Computer Engineering
Mekelweg 4,
2628 CD Delft
The Netherlands
http://ce.et.tudelft.nl/

2013

MSc THESIS

Measurement of Video Conferencing Libraries using a Test Framework based on Profiling

Maulik Prabhudesai

Abstract

Video Conferencing systems have become highly popular with the explosion of bandwidth and computing power. This has made high quality video conference possible on embedded hand held devices. This has created an ecosystem of companies developing applications which are powered by video conferencing libraries and a parallel ecosystem of companies who make those video conferencing libraries. An application developer has to design his system in a way that the video conferencing library (which is the backbone of the application) is efficiently integrated and is performing optimally. In order to do that, the developer must be acutely aware of features such as platform compatibility, resource usage, performance boundaries and bottlenecks of the library in order to make an informed decision. Hence the developer needs a test-bench which can evaluate the above mentioned features. This thesis mainly discusses the need to arrive at one such test framework followed by the framework itself.
The framework designed in this thesis consists of a network emulator which is combined with a popular network protocol analyser. It profiles the library and monitors variables essential to Quality of Service (QoS) like bandwidth usage, frame rates, frame/packet sizes and audio/video delay under varying network conditions. By observing the change in these variables, the QoS offered by the library can be determined. This information can also be extrapolated to understand the challenges in guaranteeing a minimum quality of experience (QoE).
This thesis applies the framework on one chosen library (Library 1) and Google Hangouts and discusses the results. The performance of the library under varying network conditions (bandwidth, packet loss and latency) is observed, enabling Presence Displays BV to extract information about how the library works. Presence Displays BV is building a video conferencing enabled product and uses the framework discussed in this thesis to analyse video conferencing libraries in order to build the application around it. It has been observed that Library 1 favours video quality above other parameters and only compromises video quality to accommodate more users participating in a call. Google Hangouts favour audio, motion content in the video, interactivity and robustness over video quality. This information can be used to configure Library 1 in a way such that it delivers maximum QoE in the available resources.

YOOOM

CE-MS-2013-18

TU Delft

Faculty of Electrical Engineering, Mathematics and Computer Science

Measurement of Video Conferencing Libraries using a Test Framework based on Profiling

Requirement Specific Selection and Evaluation of Video Conferencing Libraries

THESIS

submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

in

EMBEDDED SYSTEMS

by

Maulik Prabhudesai
born in Mumbai, India

Computer Engineering
Department of Electrical Engineering
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology

Measurement of Video Conferencing Libraries using a Test Framework based on Profiling

by Maulik Prabhudesai

Abstract

Video Conferencing systems have become highly popular with the explosion of bandwidth and computing power. This has made high quality video conference possible on embedded hand held devices. This has created an ecosystem of companies developing applications which are powered by video conferencing libraries and a parallel ecosystem of companies who make those video conferencing libraries. An application developer has to design his system in a way that the video conferencing library (which is the backbone of the application) is efficiently integrated and is performing optimally. In order to do that, the developer must be acutely aware of features such as platform compatibility, resource usage, performance boundaries and bottlenecks of the library in order to make an informed decision. Hence the developer needs a test-bench which can evaluate the above mentioned features. This thesis mainly discusses the need to arrive at one such test framework followed by the framework itself.

The framework designed in this thesis consists of a network emulator which is combined with a popular network protocol analyser. It profiles the library and monitors variables essential to Quality of Service (QoS) like bandwidth usage, frame rates, frame/packet sizes and audio/video delay under varying network conditions. By observing the change in these variables, the QoS offered by the library can be determined. This information can also be extrapolated to understand the challenges in guaranteeing a minimum quality of experience (QoE).

This thesis applies the framework on one chosen library (Library 1) and Google Hangouts and discusses the results. The performance of the library under varying network conditions (bandwidth, packet loss and latency) is observed, enabling Presence Displays BV to extract information about how the library works. Presence Displays BV is building a video conferencing enabled product and uses the framework discussed in this thesis to analyse video conferencing libraries in order to build the application around it. It has been observed that Library 1 favours video quality above other parameters and only compromises video quality to accommodate more users participating in a call. Google Hangouts favour audio, motion content in the video, interactivity and robustness over video quality. This information can be used to configure Library 1 in a way such that it delivers maximum QoE in the available resources.

Laboratory : Computer Engineering
Codenumber : CE-MS-2013-18

Committee Members :

Advisor: Koen Bertels, CE, TU Delft

Advisor: Robbert Smit, Presence Displays BV

Chairperson: Koen Bertels, CE, TU Delft

Member:	Arjan van Genderen, Assistant Professor CE, TU Delft
Member:	Johan Pouwelse, Assistant Professor PDS, TU Delft

This thesis is dedicated to the engineering and scientific community with the hope that after reading this, they will recognise me as one of their own.

Contents

List of Figures

List of Tables

Acknowledgements

First, I would like to thank Dr. Koen Bertels for the guidance and advice he gave me throughout my time as a master student and for the opportunity to work in the Computer Engineering group. Good ideas come from good inspiration, namely good mentors and without Koen, my journey as a master student would not have been successful.

I will be forever indebted to Omar and Robbert for their assistance and advice. The help the two of you gave me in understanding my work has been invaluable in getting this thesis finished. Both of you were very empathetic to the challenges I faced in making a transition from a student to a professional and I appreciate the effort you took for me.

I would like to thank Ashkan for going over my thesis multiple times in spite of his busy schedule and for suggesting multiple tweaks and turns in order to make it as it is today. I would like to thank Arjan for helping me throughout my time at Delft in his role as the coordinator. I was in touch with him right from the moment I applied to TU Delft as a prospective student to the day I graduated out of it. Also special thanks should be accorded to Dr. Johan Pouwelse for helping me get on track initially in my thesis and then by agreeing to be a part of my examination committee.

Thank you Akshay, Roderick, Jan Jaap and Christopher for going over my thesis multiple times providing a fresh set of eyes. Also special thanks to Amit and Prachi for helping me out with the drawings. You guys certainly helped me improve my English and the discussions I had in making my point understood actually helped me understand the problem better.

Further acknowledgements go to the wonderful people of Computer Engineering group and my colleagues namely Mark, Joost, Gustavo and Nickos. Thank you for sharing the office, the conversations we had and the help you gave me.

Not to forget my room mates and the new friends I made here who acted as my family throughout these two years lending me help and support. They made sure that I was looked after even in days when I could not contribute much due to work and other reasons. Thank you guys.

A special thank you goes to my mother. She encouraged and supported me during all my studies at Delft. Without her, all this would not have been possible.

Lastly, thank you Aaji because without you, I wouldn't have been here.

Maulik Prabhudesai
Delft, The Netherlands
21, November 2013

1 Introduction

This Chapter introduces the topic of this thesis, as well as a brief description of the challenges faced in video conferencing. First, Section 1.1 introduces the underlying problem that this thesis tackles. Then Section 1.2 explains how this thesis will work towards solving that problem. Section 1.3 discusses the contributions of this thesis and finally Section 1.4 will give an overview of the structure of this thesis.

1.1 Background

As the world progressed, newer and newer business opportunities emerged and there appeared to be a need to bridge distances and reduce travel. Deterrent economic conditions from time to time spurred a general preference for reducing travel and undertaking more and more business through remote communication. This gave a huge impetus for developing a host of communication technologies like telegraphy, telephony, electronic mail and then video conferencing. An anonymous commentator at Texas Instruments once famously quipped that every time the Arabs crank up the price of oil, the world moves closer to video conferencing.

Since its inception, video conferencing technology has evolved rapidly to enable more and more new users to get access to it every year. From dedicated conference terminals which were available with large organisations (like Texas Instruments), the technology has evolved to enable users to engage in multi-user video conferences from handheld devices like smart phones. This evolution in video conferencing technology along with improving capabilities of embedded communication devices like tablets and smart phones resulted in the creation of an ecosystem of companies developing products based on video conferencing. These products also find use in domains like medical and healthcare as people with disabilities and health issues prefer a mode of communication which does not include travel. Realising a complete working video conferencing solution and optimising it for specific use cases is highly complex and requires a high degree of technical expertise and volumes of resources. Sensing this space, a supporting ecosystem of technology companies developing video conferencing libraries has evolved. The latter companies develop the video conferencing engines which power the products developed by former companies. However in spite of the presence of so many video conferencing engines available from vendors and API developers, video conferencing is still a nebulous concept for application developers.

1.1.1 Problem Statement

The companies developing video conferencing engines make them available to application developers as Application Programming Interfaces (APIs) where a part of the function-

ality can be tweaked by playing with the API so that the engine can be tuned to product requirements. Factors like topology (which consists of design choices made at the back end), media components (like codecs, echo cancellation etc. which consist of design choices made at the front end) and algorithms which account for robustness in the face of varying network conditions are part of the core structure of the video conferencing libraries. These cannot be changed by tweaking with the API and hence it is advisable to make a careful analysis of the libraries to ensure that the quality of service and experience offered by the library matches the needs of the application.

Hence there appears to be a need for a framework which can determine performance boundaries and bottlenecks of each library. As not much information is available about the detailed design of the library, the framework needs to be able to treat it as a black box and through intensive profiling, needs to generate data pertaining to possible performance issues and bottlenecks. Armed with this data, the application developer can make better choices in system design to ensure a high quality of experience for his users. This thesis will work on developing such a framework which will provide the developer with the necessary knowledge to make informed decisions.

1.2 Thesis goals

Presence Displays BV [9] wanted to design a product which used a video conferencing engine. This thesis draws the requirements of the product and proposes a system design for it. The thesis then develops a method for evaluation of different video conferencing libraries which can be used for the design of the product. Detailed performance analysis is essential before choosing any particular library for product development. Hence, one of the goals of this thesis to develop a framework for performance analysis of video conferencing libraries.

There are many methodologies and proposed frameworks for performance analysis of video conferencing libraries which can be used to interpret the quality of experience with varying degrees of complexity and accuracy. Some aspects of this framework have been inspired by these existing frameworks. However, this framework is different in the aspect that it automates the measurements of various performance metrics, reducing assessment time and complexity but also to some degree, accuracy.

The framework can be used to do qualitative assessment of quality of experience based on quantitative results. The framework does not aim to provide software performance bottlenecks but to assess the performance of the system as a whole from a black box perspective. This kind of testing provides a complete and a true assessment of performance and also provides quicker insights into the limitations of a library. This knowledge helps the application developer to make faster and more accurate design decisions and aid rapid software development.

1.3 Thesis Contributions

This thesis contributes a framework which can be used for quantitative analysis of the performance of a video conferencing library. This information can be interpreted to

predict qualitative information like the quality of a video call. The thesis also discusses experiments conducted using the framework which provide insights into the performance and bottlenecks of a video conferencing library and promulgates our understanding of how video conferencing libraries work in general.

1.4 Thesis organisation

This thesis is organised as follows: Chapter 2 will discuss a brief overview of video conferencing technology, topologies and different components needed in video conferencing system design. The problem at hand which led to design and development of the framework talked about in this thesis is discussed in Chapter 3. In Chapter 4, the requirements which a video conferencing library used for the development of the product are discussed. This chapter also includes a survey of available video conferencing software development kits (SDKs), or APIs which can be used for potential development of the product. Chapter 5 discusses the design of the framework and what components are used in its design. The framework was applied to some video conferencing libraries and existing video conferencing software products. The set up of the experiments using the framework and its results are discussed and analysed in Chapter 6. This thesis is concluded in Chapter 7.

Video Conferencing Technology and Considerations

2

2.1 Introduction

The way people communicate with each other has changed vastly over the years. Letters gave way to telephones. Later letters changed into emails and telephone turned into internet telephony - Voice Over Internet Protocol (VOIP) which further evolved into video conferencing (VC). Video conferencing has a history of about 40 years with the first video conferencing prototype deployed commercially by AT&T Corporation sometime in the 1970s. Video conferencing is basically transmission of audio and video to and from two or more geographically different locations in a way which simulates a real physical conversation. Video conferencing is done using two approaches [53]:

1. Utilizing high-quality VC equipment with dedicated resources
2. Implementing a VC application on personal devices like smart phones, computers etc.

Video conferencing technology started using the first approach as an expensive technology which needed dedicated bandwidth and expensive dedicated equipment. Since then a lot of the technology has been standardised and video conferencing is evolving into more of a plug and play technology. Today, bandwidth explosion has made approach 2 possible. Applications like Skype, Yahoo Chat, Google Hangouts which support multi way audio/video conference are fairly common. Due to increasing number of users opting for voice/video communication over the internet, the share of time sensitive multimedia data of the share of total data has increased. Video conferences are still bandwidth intensive and perform poorly in low bandwidth scenarios (e.g. GPRS). This coupled with the high networking costs makes video conferencing still a largely unpopular tool. This chapter looks at essential modules which go into a video conferencing system and goes through existing architectures and solutions.

The earlier video conferencing systems were analogue in nature and were simply two video links in two directions. This technique was as old as the television. Evidence suggests that they were used in the 1940's when the German post office started video telephony between Berlin and various cities. NASA famously used UHF and VHF video links in two directions for manned space flights. This technology was very expensive, inefficient, required high power, heavy equipment and could not be used by the ordinary population. These techniques used the first approach and were still primitive as compared to video conferencing applications in use today. As research into digital audio and video communication picked up pace, video conferencing started to become more and more commercially viable. Research into codecs, hardware, software and network properties made video conferencing technology accessible to ordinary people on their PCs.

In the 1990s, video conferencing was fully integrated into internet with IP based video conferencing. It was the decade of 2000 where free applications like Skype, Yahoo along with Google's plugins powering applications like Hangouts made video conferencing possible on nearly every PC present on the planet with sufficient bandwidth. Now as video compression techniques are becoming more and more efficient and the bandwidth is ever increasing, attempts have been made to achieve high definition video conferencing [11]. In the decade of 2010, newer technologies have made high quality video decoding/encoding on embedded computing platforms possible. This means that mobile phones and tablets can now be connected to the internet and video conferencing can be made possible on them. This is made evident by the number of video conferencing applications like Skype available freely [75].

In the subsequent sections, various concepts related to video conferencing are introduced. It is also discussed why video conferencing is different compared to other real-time video applications like video on demand. In spite of differences, the core essence of video conferencing is real-time delivery of inelastic data and challenges faced in real-time video delivery in general will be discussed.

2.2 Video Conferencing Technology

Video conferencing is a lucrative market with many opportunities and this makes the business highly competitive. Video conferencing software vendors take great care in optimising each and every module in the software to the fullest extent. Thus details of modules which go into making such libraries are looked at. Lastly video conferencing is discussed at the system architecture level.

But first it is important to have an overview of the technology which is essential for realising a video conferencing system. In the following sections, VC technology implemented on PCs is discussed primarily and not using dedicated VC equipment. To realise a video conferencing system, a PC needs:

1. Video capturing hardware (Camera) - The camera captures live events occurring at the source for the receiver to see in real-time. The quality of the camera heavily determines how the video will appear at the receiving end. A good camera fetches images at 25-30 frames per second and a minimum resolution of 640 x 480 pixels.

2. Sound capturing hardware (microphone) - The frequency used in video conferencing usually does not exceed 7Khz and hence any capturing device which supports that frequency is sufficient.

3. Video illustration hardware (a screen) - The receiver sends its video to the source and it is displayed at a screen at the source. Video resolutions are fixed for certain standards e.g. The H.323 standard supports 352 x 288 pixels.

4. Sound generation hardware (a speaker) - A normal speaker is sufficient to realise a video conference.

5. An underlying real-time data delivering system (e.g. Real-Time Transport Protocol - RTP)

6. A PC (or alternatively an embedded device) which packs all these components together which is connected to the internet

These were the hardware requirements which a PC (or an embedded platform) needs to satisfy in order to run a video conference. A typical video conference application needs many modules of software namely a user interface layer, a session management/signalling layer, a media module and real-time network stack. This can be seen in the figure 2.1

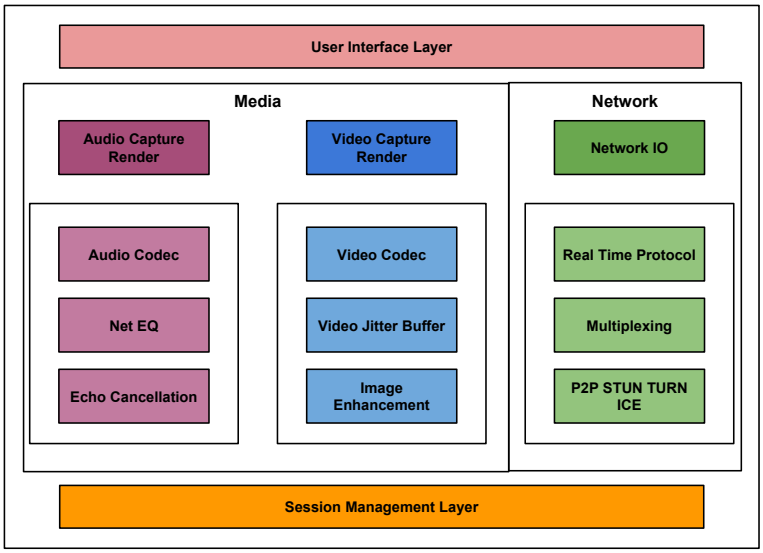


Figure 2.1: A typical video conferencing system architecture based on WebRTC [10]

The user interface layer is where the user interacts with the software by means of a platform for video rendering and interaction with participants in the conference. The session management layer is responsible for signalling, conference initiation, and conference management at the technical level. The media module is responsible generating content (audio and video) from the users and packing them into packets which can be sent over the internet. It also renders the audio/video data back on the screen for user's consumption. The media module contains various audio codecs, video codecs, text codecs and other modules essential for maintaining good quality of audio/video, like equaliser for audio and image enhancement techniques for video. Apart from that, there are some modules unique to a video conference like echo cancellation modules.

The network stack contains real-time protocols which send inelastic data piggybacked on UDP packets. Popular protocols are Real-Time Transport Protocol (RTP) [72] and Secure Real-Time Transport Protocol [21] (which encrypts data for security). These protocols, in combination with Real-Time Transport Control Protocol (RTCP), [72] also send information which is used for interpretation of the media (information about audio/video codecs, change in frame rate, bit rate etc).

Section 2.3 describes each component in the video conferencing stack in detail.

2.3 Video Conferencing Stack

This section discusses various components which go into the software stack of a typical video conferencing system.

2.3.1 User Interface Layer

As any video conferencing system is designed to facilitate communication between two users, the software stack must contain a layer which interacts with the user. A user typically checks if his partner (with whom he intends to talk) is online or whether he is willing to accept the call before initialising the call. Neither does a receiver want to start a video conference without his consent (due to privacy issues). Also the users might feel the need to disable audio or video in the duration of the conference or change camera or speaker settings. Hence there has to be a user interface through which the users can control a conference.

2.3.2 Media

As shown earlier, real-time video delivery imposes strict timing constraints. For an interactive application like video conferencing, the end to end latency should be kept under 300ms for the worst case of acceptable performance [53]. This means that from the moment a frame of video is captured at the source's camera, encoded into bits, constructed into packets, sent over the internet, received by the receiver, reassembled into useful packets, decoded into playable video and is displayed on the screen of the receiver, the total time elapsed should not be more than 300ms. The delay in the network is attributed to the number of routers and the traffic density (and congestion) in the link. This delay is thus unpredictable and a comfortable margin of error is maintained to account for these delays. Also video is typically encoded before sending over the internet to save bandwidth. Thus the time spent in encoding the video should be kept as minimal as possible.

Video Codecs

A successful video conferencing system has an efficient way of encoding and decoding data. Even a standard definition frame costs a lot of storage space and sending such data over the internet for real-time delivery is not recommended. Hence the systems use a coding/decoding algorithm (codec) to compress video. Video encoding is a non trivial task and is highly computationally intensive. The camera captures individual frames at a certain frame rate and then it is the job of the video codec to carefully compress the video so that maximum quality should be transmitted in minimum space.

Video clips are made up of individual frames. Video coding attempts to code each frame and the subsequent frames such that maximum compression is achieved. Each frame is divided into small blocks and the blocks then converted from the spatial to frequency domain using a discrete cosine transform. Then the frame in the frequency domain is quantized for different frequency values. The human eye is less sensitive to higher frequencies and thus higher frequencies are coded into a smaller number of bits

and vice versa. If the frequency values are encoded into a larger number of bits, the video quality turns out to be better at the cost of more data. Many codecs tweak this parameter to alter video quality on the fly. As subsequent video frames are similar to each other, video codecs aim to predict a future frame from a previously encoded one (or alternatively a past frame from a previously encoded future frame).

Motion estimation is performed where a number of previous frames are checked for which frame matches the best with its prediction. However prediction algorithms rarely predict a frame with 100% accuracy. However the difference between the actual and predicted frame is quite small compared to the frame itself and hence only the difference is encoded. The frames encoded independently are called I frames and frames encoded using an I frame is called a P frame or a B frame depending on the temporal co ordinate of the I frame. Hence, if the receiver has an I frame, the loss of a P frame can be easily compensated. Periodic I frames cause greater data rate but ensure higher fault tolerance at the receiver. This part is called motion estimation and it increases the efficiency of coding upto 200:1 for some codecs [22]. Some codecs, particularly the H.264 employ various image enhancement techniques (they might be implemented independently also) to correct various effects like blocking and ringing which occur because of division of the image into blocks for processing. This part is also the most time consuming and demands a huge amount of resources/processing power from the hardware. In general, video codecs take up a huge amount of processing power and hence require speed up via GPUs or dedicated hardware.

It is widely known that the H.323 standard of audio/video telephony supports H.261 [81], H.263 [35] and H.264 [73] codecs. Of them H.264 is a video codec jointly developed by MPEG (Moving Pictures Expert Group) and ITU (International Telecommunication Union). It is widely used for high definition video broadcast over the internet and is an industry favourite for encoding videos. As it is a standard, dedicated H.264 hardware is found on many embedded computing platforms including Google Nexus 7 and 10 which makes video generation/decoding easier.

However newer and newer codecs are being developed every day and hence from an application point of view it is important that the software architecture is modular so that a codec may be easily replaceable with a newer more efficient one. This is evident from the popularity of the SIP protocol which allows the users to decide their own codec implementation as compared to H.323 which supports only 3 codecs.

The other codecs which are fast gaining popularity are the VP series of codecs from ON2 technologies. The VP range of codecs are highly successful as VP6 has been used in Adobe Flash and VP7 being used in Skype [87]. In late 2008, VP8 was developed as a competition to the H.264 standard. VP8 is quickly finding popularity with video conferencing application developers and Skype has famously declared that progressively both codecs (VP8 and H.264) will be used in various applications.

VP8 has been designed keeping in mind implementability over various platforms, and has been tested to prove real-time encoding/decoding ability on relatively low end embedded hardware. VP8 has also been shown to cost less number of processor cycles to decode as compared to other algorithms. More and more embedded platforms are moving towards multi core systems as is evident by the launch and success of various tablets like Google Nexus 10 and phones like HTC One X. The inability of codecs to

be parallelizable hinders with their implementation of such platforms and VP8 has been designed to exploit multiple processor cores. Many of the smaller algorithms which are a part of a larger codec algorithm cannot be parallelised and VP8 uses a modified version of these algorithms which can be parallelised without compromising on efficiency [45].

In a comparison with its main opponent, H.264 edges ahead of VP8 because it enjoys hardware acceleration support from various system on chip (SoC) manufacturers. However, newer and newer vendors are switching to offer hardware acceleration of VP8 as is evidenced by the launch of Samsung Exynos 5 chip [84]. Other manufactures are starting to provide devices integrated with VP8 hardware support. As VP8 is open source free technology, its usage is expected to save money paid in royalties to MPEG for the use of H.264. Thus for a video conferencing library, usage of VP8 or design keeping in mind easy integrability with VP8 augurs well for the future of the application.

Audio Codecs

Audio codecs are programs which are capable of encoding and decoding audio digitally. Some famous examples of audio codecs are Opus, iSAC, iLBC and MP3. Opus codec is an audio codec capable of supporting constant as well as variable bit rate encoding with bitrates varying from 6Kbit/s to 510 Kbit/s. The frame sizes vary from 2.5 ms to 60 ms and the sampling rates vary from 8KHz to 48KHz which enable it to reproduce the entire range of the human auditory system. iSAC is a wideband/super wideband audio codec developed by Global IP solutions. It is bandwidth adaptive and robust and is used in millions of VOIP implementations with an adaptive and variable bit rate. iLBC is a narrow band audio codec also developed by Global IP solutions used in VOIP and streaming applications. Media formats like codecs and other media modules evolve over time and the ideal library should be flexible and modular enough to allow integration of newer and more efficient media standards into it.

Echo Cancellation

Because of the interactive nature of video conferencing application, an important factor which may affect the quality of experience (QoE) is the echo in the system. Audio generated at the source is generated at the speakers at the receiver and is also recorded at the microphone of the receiver and sent back to the source as a part of the receiver's audio feed. If unchecked, this causes the sender to hear his own voice again. Subsequent echoes multiply and sound reverberates in the audio channel rendering it ineffective for any form of audio communication. The microphone may pick up some noise at the input and due to the positive feedback loop created by the echo, a howling sound may be observed rendering the video conferencing system useless and causing irritability to the users.

Echo cancellation technology has been needed ever since the first interactive audio communication devices came into being. Old telephony systems handled this challenge by placing attenuation blocks on the channel to act as echo suppressors. As digital telephony developed, newer and newer algorithms like Least Mean Square (LMS) and Normalised Least Mean Square (NLMS) amongst many others have been developed. This module is fundamental to the realisation of a successful video conferencing system

and operating systems such as Android have a built in echo cancellation module [15]. Upcoming IETF standard WebRTC uses AEC software developed by Global IP Solutions which was bought by Google [10].

2.3.3 Network Stack

This section describes in detail the components which go into the network software stack of a general video conferencing system.

Real-Time Transport Protocol

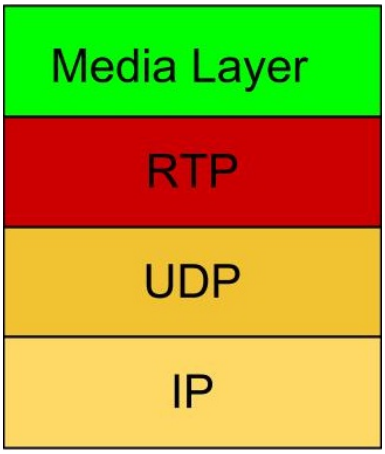


Figure 2.2: Real-Time Transport Protocol Stack

Real-Time Transport Protocol (RTP) is a transport protocol which provides end to end connections for transmitting real-time media packets like audio and video. RTP was standardised by the Audio Video Working Group of the Internet Engineering Task Force in 1996 [72]. Real-time media may be either a stored video/audio which is demanded at a remote location over the network (e.g. video on demand), or it may be generated in real-time like a live video conferencing application. In the case of video on demand, a delay of few seconds is tolerable while in the case of video conferencing, which is interactive in nature, a delay of not more than a few hundred milliseconds is tolerable.

In common protocols of the transport layer like Transmission Control Protocol (TCP), the retransmission delays make it unsuitable for sending real-time media. Media applications are typically error tolerant and thus retransmission brings unnecessary delays in ensuring reliability which is not needed for the application. UDP (User Datagram Protocol) may be used, but UDP is a best effort delivery service and packets may be delivered out of order thus making it unsuitable for sending real-time media. This dilemma led to the design of RTP.

RTP provides a standardized format for delivering media packets through underlying transport and network layers. RTP can be used with any of the protocols of the transport layer. However RTP is commonly used with UDP and not with TCP because of the aforementioned delays associated with TCP. However when piggybacked on UDP, RTP is

a best effort delivery service and blind to congestion in the network (congestion detection feature is associated with TCP). Thus RTP has it's own congestion detection strategy for media applications which it implements through Real-Time Control Protocol (RTCP). The control protocol's functionality enables it to detect congestion and provide feedback on Quality of Service (QoS), membership (number of participants in the conference) and loop detection, meaning that it can detect if a packet was forwarded back to its source from some other participant in the conference.

In this section we will see why RTP is successful in ensuring real-time delivery. An RTP packet can be seen in the figure 2.3.

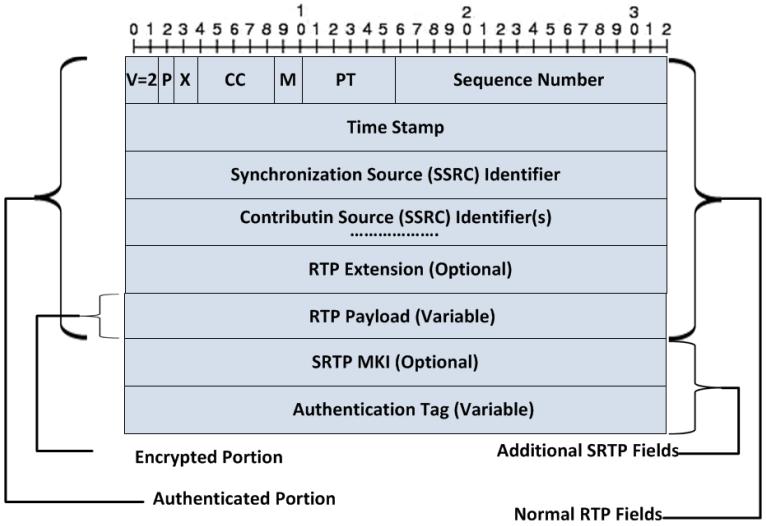


Figure 2.3: An RTP Packet [3]

The unique fields in an RTP header are:

- **Payload Type** - The PT field helps the application determine whether the payload is audio, video or some other real-time data. The payload type helps specify the codec, the encoding in case of audio codecs and similar functions. The field supports 128 different options (7 bits) and many of the options are not utilised. Hence this protocol is open to standardizing newer and newer audio and video codecs.

- **Sequence Number** - This field is used to indicate the sequence of the incoming packet and helps the application detect packet loss. Packet loss as we will see is helpful in determining the congestion in the network and that information will be subsequently used to modulate the data rate at the source. But generally, RTP packets are piggybacked on UDP, and UDP causes segmentation and reassembly of packets. This field helps the application layer re-arrange the packets in the correct order.

- **Timestamp** - The time stamp field differentiates RTP from TCP or UDP as the time stamp adds a temporal coordinate to the data which is usually spatial. In

real-time media applications, a packet which reaches after its intended display time is useless. This field helps the application to detect and discard useless packets. This information is also used to determine congestion and to modulate source data rate.

- **SSRC** - This field is the Synchronisation Source Identifier field. This field is used for timing synchronisation for a session and is crucial for maintaining the real-time aspect of a session.

For providing secure payload delivery, a specific profile of RTP called Secure RTP (SRTP) is used. The audio and video packets or packets from different RTP sessions may be multiplexed into one single RTP packet too [32].

To ensure QoS for real-time video delivery, the following issues have to be handled:

1. **Bandwidth** - There has to be a minimum bandwidth available at all times during the transmission.
2. **Delay** - The end to end delay has to be bounded and within a minimum value (300 ms for video conferencing)
3. **Data Loss** - Packet loss ratio must be kept below a certain threshold for ensuring quality of service.

Congestion occurring due to an unreliable network connection is the primary cause of decrease of Quality of Service for real-time video delivery as packets are often lost or delayed due to decreased bandwidth emanating from high network traffic. Thus any efficient real-time video delivery system, whether for video on demand or video conferencing, should aim at developing a robust response towards congestion in the system.

Congestion Control

The challenge of any video conferencing applications is transmission of real-time video over constraints such as limited bandwidth and delay. The latency is a very critical component which will decide if the system is usable or not. Latency is the time lapse between the time the video appears on the source (capture equipment) to the time the video appears on the destination (rendering hardware - screen). For an interactive application like a video conference, the end to end latency for the system should be less than 300 ms. The lower the latency, the more room there is for error resulting due to unreliable network connections and sudden unexpected congestion in the network.

The main difference between real-time traffic and non real-time traffic is that non real-time traffic is elastic and the data rate can be changed to match the congestion in the network. Thus it is sent using TCP which ensures quality of service by transmissions and retransmissions. Real-time traffic however does not enjoy that luxury. It can tolerate packet losses than have entire packets retransmitted by TCP again.

As UDP does not have a congestion control strategy unlike TCP, an external congestion control strategy must be used. Then depending on the congestion, the bit rate

of the video is varied. By using variable bit rates and a send buffer, non elastic real-time data is made partly elastic and tolerant to a certain amount of congestion at the expense of lowering the quality. Also because of the strict latency constraints imposed upon video conferencing applications due to the interactive nature unlike video on demand applications, evaluation of congestion and changes in the bit rate have to be done dynamically.

Congestion control strategies can be broadly classified into either rate control, rate adaptive video encoding or rate shaping [89]. Rate control follows a TCP like window based congestion control approach. This is further divided into rate control at the source and rate control at the receiver. In the first approach, the source varies the rate of the sent video to combat congestion. This can be done either by probing the channel for packet loss at periodic times or by evaluating a congestion by using a mathematical model. Some of the strategies use a TCP like approach for congestion evaluation using the formula [36]. Depending on the nature of congestion control required, other parameters like quantization rate, sampling rate, encoding rate may be changed to change the bit rate of the video generated at the source.

$$\lambda = \frac{1.22 \times MTU}{RTT \times \sqrt{p}} \tag{2.1}$$

The above equation gives the bandwidth λ of a TCP connection in the face of packet loss in the network. MTU is the maximum packet size used in the connection by the underlying protocol, RTT is the round trip time and p is the packet loss ratio observed in the network. This formula is used to develop TCP-like congestion control for UDP or media flows which is TCP friendly.

Congestion control can also be implemented at the receiver with the help of special codecs which encode the video into multiple layers. This technique is called multiple description coding (MDC) or Multiple Layered Coding (MLC). In multiple description coding, the video is encoded into many descriptions of varying bit rates and qualities such that any one description is independently capable of reproducing video of an acceptable quality. The receiver subscribes to as many layers as possible depending on its bandwidth. The more descriptions available with the receiver, the higher the video quality it is able to reproduce.

MLC is similar to MDC except that the layers are categorized into base layer and subsequent layers. The base layer is mandatory for decoding a video of acceptable quality and if additional layers are present higher video quality can be obtained.

In a typical conference, different users may have different link qualities and available decoding hardware. A higher video quality may be perfectly acceptable to one receiver whereas some other receiver may need a lower quality. This aspect makes it particularly difficult to implement multicasting of video packets from the source. Congestion control at the receiver works better as it gives different receivers the power to choose what video is suitable given their network conditions and hardware. Different layers are sent as multi casts to different groups and each receiver subscribes to one or more groups depending on the nature of congestion (and also other criteria like the available decoding equipment). If congestion is not detected for a long time, the receiver subscribes to more groups to obtain higher quality. When congestion is detected, the receiver drops some layers to

ensure smooth degradation in receiver quality. The receiver may also use a mathematical model to determine congestion and decide upon the number of layers to subscribe to.

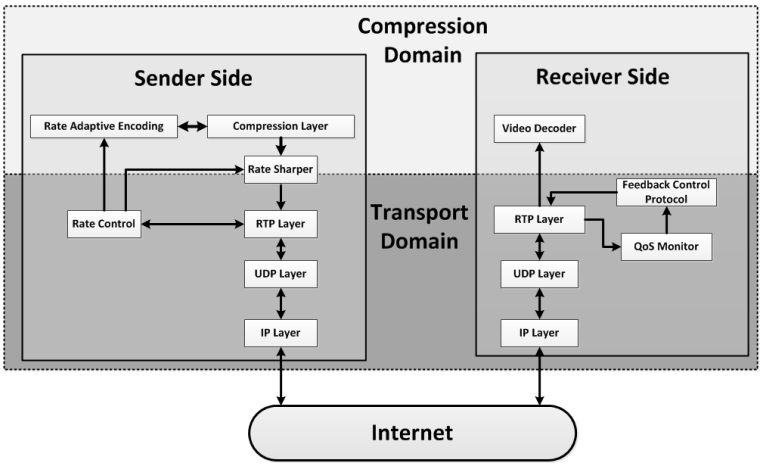


Figure 2.4: RTP Usage and Congestion Control [89]

Figure 2.4 describes how RTP and RTCP together formulate a congestion control strategy for real-time video streaming [89] and Figure 2.5 describes a model of Skype's video rate adaptation scheme. As it can be seen in the figure, the sending rate $r_s(t)$ is decided using three parameters, namely frame quality $q(t)$, video resolution $s(t)$ and frame rate in fps (frames per second) $f(t)$. Information about video quality received at the destination is sent back to the source using a feedback channel. RTCP may act as this feedback channel when the transport protocol is RTP. RTCP sends various feedback messages requesting a new I frame due to loss of decoder context, informing the source about loss of picture or loss of a picture slice. The feedback messages may also ask the source to do a temporal/spatial trade off or to limit the video source rate under a particular threshold. The encoding parameters of the video are varied depending on this feedback.

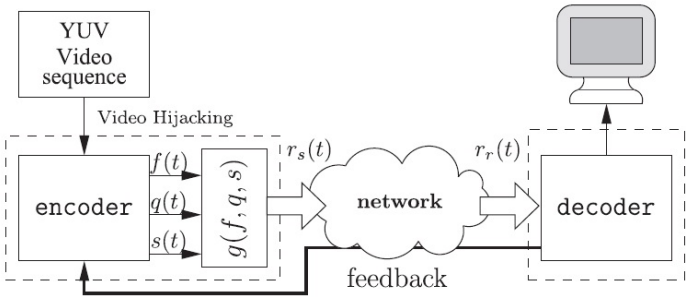


Figure 2.5: Model of the Skype video rate adaptation scheme [30]

The third approach to congestion control is rate shaping [89]. Rate shaping is dynamic adaptation of the rate of media packets sent out in the network. If congestion is

detected, the sender may randomly drop a few frames. Those frames can then be reconstructed at the receiver using golden frames. This actually increases video quality and at the same time, reduces congestion. Alternatively, the bitrate of the frames may be changed by changing the encoding scheme or the motion vector. The human eye is not sensitive to higher frequencies and higher frequencies can be encoded into lower number of bits or dropped completely in the quantization process.

The Figure 2.6 shows a brief overview of a video conferencing system in a browser using RTP [10]. The audio and video data is generated separately. The data is assembled into packets and sent over the internet using the Real-Time Transport Protocol. The different pipes for both audio and video signify that separate SIP sessions are used for audio and video. Thus quality of experience can be guaranteed separately for both audio and video streams. If the video session has to be terminated for lack of bandwidth or for other reasons, the audio session can continue without delays in setting up the session. As we have seen earlier, RTP packets or different sessions may or may not be multiplexed. The RTCP pipe is used for conveying congestion information which is used by the application to alter the data rate. The audio and video packets are displayed at the receiver's end taking into consideration their time stamp from the RTP packet.

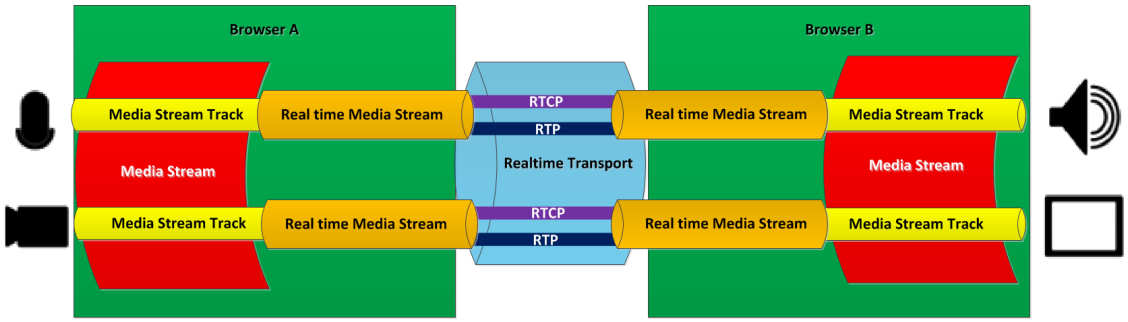


Figure 2.6: A video conference system at work [10]

Network variables like packet loss ratio, latency, change in RTT over time (varying RTT is called jitter) vary for different network conditions. It can be concluded that the goal of any congestion control strategy is that any developer should be able to customise the response to changing network quality in order to be able to control user experience to a much larger extent.

NAT Traversal

Majority of the users engaging in video conferencing are behind some sort of Firewall or a NAT. A drawback of NAT is that NAT makes it difficult to deploy new IP applications. NAT basically breaks end to end connectivity and hence some sort of NAT Traversal technique is needed to realise client to client networking applications - like a video conference. Session Traversal Utilities for NAT (STUN) are a set of methods used by many protocols for NAT traversal. Protocols like Interactive Connectivity Establishment use STUN servers (which are located in the public internet domain) to establish connectivity behind a NAT.

Error Control

In any form of data transmission over the internet, packets are lost and information is corrupted. The lost packets are usually resent over to the receiver. This is perfectly acceptable if the data has no sense of timing (large files or text based information). However, in the case of real-time video delivery (both video on demand and video conferencing) resending the data cannot be the preferred solution because data which arrives late is meaningless. Hence real-time video delivery systems employ a number of measures for error correction, error resilience, error concealment and also retransmission (only if the delay is within acceptable limits).

Forward error correction is a preferred strategy for error correction [89]. A block of k packets is encoded into a block of n packets where $n > k$. Assuming that some packets are lost in transmission, the entire sequence of k packets can be recovered if the number of received packets K is such that $n > K > k$. This provides tolerance to errors at the expense of increased transmission rate and delay. Another method is encoding some redundant information about the previous block in the current block so that the previously sent block can be reconstructed in the eventuality that it is lost. Or a combination of these techniques can also be preferred depending upon the network characteristics. If the link quality between the source and the receiver is relatively good and the Round Trip Time (RTT) is less, a receiver might request a retransmission or a sender might retransmit packets if the time to display the packet is sufficiently large.

2.3.4 Session Management in Video Conferencing

From the conference perspective, a chat (one way or multi way) is initiated by the user. But at the protocol level, establishing, managing and terminating chat sessions is done using session management protocols. Session management protocols provide application level layers the necessary information to initiate, maintain and terminate a conference. Two of the most popular protocols used for session management are Session Initiation Protocol (SIP) [74] and the H.323 [78] standard for session management. Figure 2.7 shows how the session management protocols fit into the Open Systems Interface (OSI) [76] hierarchical layer:

The user agent is the application level software which is running and managing the video conference. The lower layer is the media generation layer where media may be represented by either audio, video or even text. The session management requirements are independent of the nature of media and because of this, session management protocols are independent of lower layers like the transport layer, network layer and the physical layer of the OSI stack.

Session Initiation Protocol

Of the two major protocols that are being discussed, SIP (Session Initiation Protocol) [71] is an Requests For Comments (RFC) [47] standard from IETF (Internet Engineering Task Force) [23]. SIP is open and allows users to modularly choose a wide range of protocols running on layers on top as well as under it. SIP also allows users to build hooks into it to add application oriented functionality [24]. SIP leaves network level

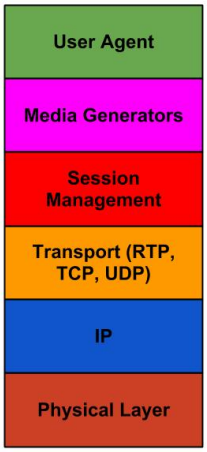


Figure 2.7: Session management layer within the OSI layer

reliability issues to be solved by the network level protocols itself. This means that SIP is very light compared to other session management protocols like H.323 and also highly modular. This highly modular structure allows users to integrate the SIP with other standards developed for specific functions. SIP messages are text formatted like HTTP (Hyper Text Translation Protocol) [34]. Text based messages are also easier to debug and hence this standard is gaining fast popularity across developers all over the world [77].

SIP works in the following way: A SIP user agent initiates a session request. The client application level software is called the SIP user agent. The user agent can work as a client (when requesting a connection) or as a server (when responding to a request). These roles are temporary and when the roles are switched, user agent client and server positions are interchanged. This request is forwarded to the other user agent who is going to be a part of the conference call. A SIP server receives this and returns either an accept or a reject response depending on whether the user wishes to take the call or not. Proxy servers are redirecting servers which are used to route the call request to its proper destination. Once the request procedure has been done, the call establishment procedures take a few handshakes before the media exchange occurs. Figure 2.8 describes the call flow diagram of SIP.

H.323

The other popular choice for session management, the H.323 protocol was standardized by the International Telecommunications Union and was designed to provide sessions on any packet network, including a telephone line. It is a comparatively complex and a pre-agreed upon standard and hence is deterministic and does not allow users to make

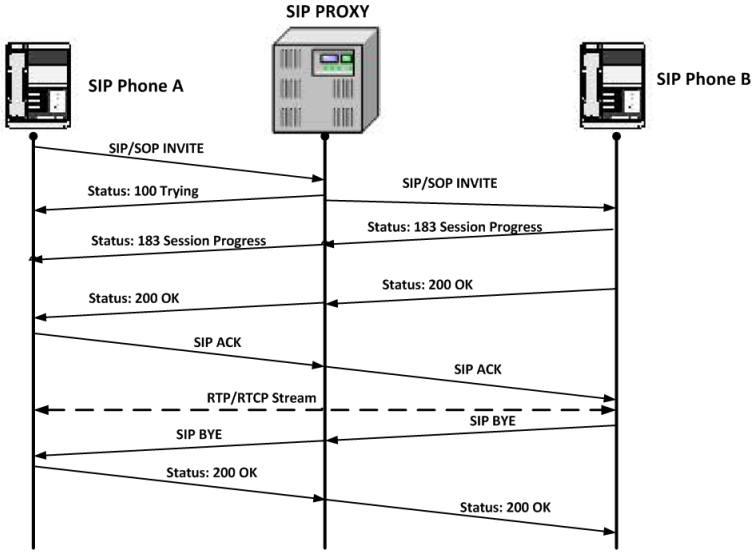


Figure 2.8: SIP call flow diagram [57]

many application specific modifications. H.323 has a focus on providing a bare minimum quality of service and hence specifies everything from codecs to the underlying transport protocol). It has various components aimed at different use cases for e.g. H.245 for multimedia call set up and to provide feedback about quality of service, T.38 which is a text codec used for facsimile, T.120 which is a set of data protocols for media conferencing etc.

It assumes and provides for various cases when the underlying network may develop issues. Extensions in H.323, due to the lack of a general modular structure as compared to SIP, are typically non standard. This makes integration across varying standards difficult. Unlike SIP, H.323 has binary messages which make debugging difficult. As compared to the simplicity of SIP, the complexity of H.323 protocol stack can be observed in the following figure 2.9. As it can be clearly seen, there is a lot of overhead to support functions, many of which may not be needed for a particular application. Also the support for codecs in this format is fixed which means that newer and more efficient codecs cannot be easily integrated into this format.

Other Protocols

The other widely used protocol for session management is the Jingle protocol which was developed by Google. Jingle is an extension to the Extensible Messaging and Presence Protocol (XMPP). XMPP is famously used in Google Talk and Google Hangouts.

Session Description Protocol (SDP) may be used with Session Initiation Protocol for describing Session Initialisation Parameters. A session may be described by specifying identifiers like network address of the originator, encryption key, port numbers of ports used in connection or sending real-time media and the time the session is active. It may be used to describe nature of media like codecs, type of protocol for media delivery, nature of media interaction (audio, video, text, presence etc) and the uniform resource

locator (url) of the session if available. SDP is being proposed for use in the upcoming WebRTC [44] standard of video conferencing and also features in many other video conferencing applications using the SIP. However many applications (famously Skype) do not use the SDP at all.

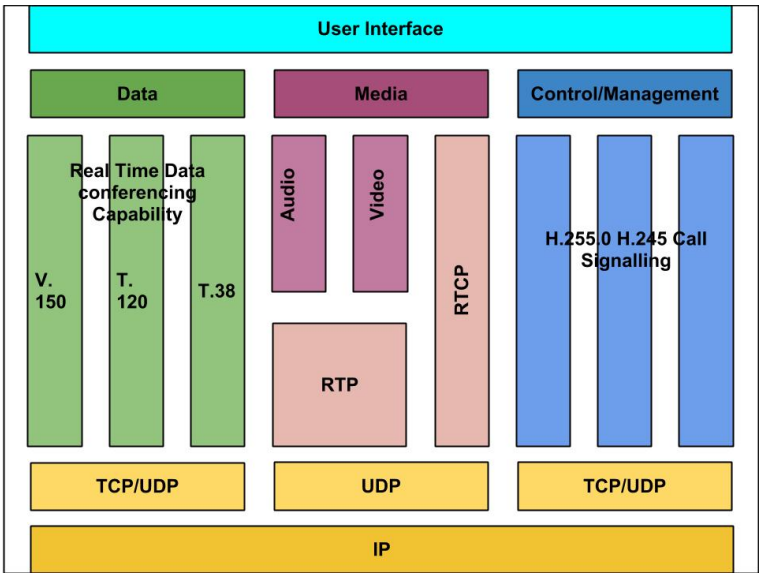


Figure 2.9: H.323 Protocol Stack

2.3.5 A case study: WebRTC

WebRTC is a free, open source client side library which enables Real-Time Communications (RTC) capabilities via simple Javascript Application Programming Interface (API) [10]. The library also has modified versions which enable application development on Android, Windows, and iOS. The WebRTC API allows third party developers to develop real-time communication applications like a video conference. The library contains basic building blocks for high quality audio/video communications in real-time. The library is being standardised at both the API level (by the W3C) and at the protocol level by the IETF. WebRTC is currently a W3C working draft and is proposed to reach candidate recommendation status in a year [83].

The transport engine (network stack) of the WebRTC library uses RTP and RTCP for real-time transport delivery and also employs NAT punching technologies like STUN, ICE and TURN (Traversal Around Relays around NAT). This enables WebRTC to ensure connectivity even when one user is behind a firewall or a NAT. WebRTC uses dynamic jitter buffers and error concealment techniques to counter packet loss emanating from unreliable networks.

The Voice engine supports various codecs like iSAC, iLBC and Opus. The audio package also includes other useful functionality like acoustic echo cancellation module developed by Global IP solutions and modules for automatic gain control, noise reduction and noise suppression. However other audio codecs can be integrated with the WebRTC

library after some amount of engineering effort. The video engine of WebRTC mainly supports the VP8 codec but as with the case with audio codecs, newer codecs can be integrated with somewhat engineering effort. The video engine also has modules to enhance image quality and a buffer to conceal packet loss and to ensure smooth video quality modulation in the presence of rapidly changing bit rate. The components of the voice and video engine are supported over multiple platforms and their highly modular structure gives developers the freedom to choose their own implementations and customisations.

However as the standardisation process of the API is still quite young, a lot of implementation specific concerns are unadressed and it is up to the developer to solve them locally. WebRTC is designed mainly as a client side API designed for peer-to-peer telephony. However it has been discussed in previous and subsequent sections that the peer-to-peer choice for architecture is not very bandwidth efficient for multi-user video conferencing and poses a lot of challenges in the case of heterogeneity in bandwidth and users. Also WebRTC has no server side functionalities, in case a media relay server has to be implemented. Hence implementing a media relay server and integrating it with a WebRTC powered client is left for the developer to figure out. Apart from that, WebRTC has no specifications for signalling services and it is left up to the developer to decide which signalling protocol to use for his application.

WebRTC uses a unique congestion control algorithm called RTP Media Congestion Avoidance Technique (RMCAT) which is designed to compete with both long standing and bursty TCP flows, other RMCAT compatible and non compatible media flows and even LEDBAT flows [64]. Finalising a congestion control strategy for WebRTC is part of the standardisation effort which hopes to have a single robust congestion control strategy for all WebRTC real-time interactive media flows.

WebRTC also carries a Berkeley Software Distribution (BSD) licence and can be used to build commercial products and copyright them. Thus the WebRTC package is a unique, free, high quality complete solution that enables real-time communication over the internet. There are many contemporary solutions available which use WebRTC as a base like OpenTok [62], AndBang, Plivo [67] and Twilio also uses WebRTC in its video conferencing client. WebRTC has thus made cross device video conference easy [70] across Tablets, phone, desktops, laptops and even Google TV.

2.4 Topology

The software stack only comprises of the front end of any video conferencing system. The topology comprises the back end of the video conferencing infrastructure. Just as the front end, there are multiple design choices available at the back end to realise an infrastructure complete with benefits and disadvantages of each option. This section discusses the various topologies which can be used for a video conferencing architecture, namely client-server, peer-to-peer and peer-to-peer with peer contribution towards video distribution.

2.4.1 Client-Server

As we have discussed, one of the main bottlenecks in providing a good video conferencing service is the bandwidth. The users may be spread over a varying bandwidth conditions and even then, the upload bandwidth is typically one-fifth of download bandwidth. This means that multiple copies of the same video cannot be sent over the limited bandwidth more number of times. This bottleneck multiplies in the case of high definition video and as a high definition video has a higher data rate than standard definition video.

The other concern with making direct connections with the other users for video conference is that the receivers among themselves have great heterogeneity in network conditions (namely bandwidth) and hardware. A video quality which may be enjoyed easily by one receiver cannot be decoded or downloaded in real-time at the other. To overcome this, the sender must encode video in different bit rates/definitions which may be decoded at the receiver. Now as we have pointed out in earlier sections, encoding a video is a very complex task which demands huge resources. Encoding the same video with different bit rates is complex from the source point of view. Alternatively video may be encoded in a format which can be decoded by the weakest of the receivers but then in the case of a group video conference, the quality of experience of an entire group may be severely degraded by presence of a weak peer. In cases where the underlying network supports multicasting, it is an easier alternative. This approach saves upload bandwidth as one video packet (decodable by the weakest receiver) may be sent across all users. This approach is again unjust to stronger peers but saves bandwidth as different connections do not have to be formed with each of the peers. However for multicasting to work effectively, all routers between the source and the receiver have to be multicast enabled. Multicasting is not popular among internet service providers because deploying multicasting routers is not only highly cost intensive but also time consuming. This creates a security risk as the system is vulnerable to attack.

The architecture which compensates for heterogeneity in bandwidth and hardware is the client-server model. Each client uploads a video to the server and downloads from a server the feed of each of the users in the video conference. Thus each user has to upload only one stream and download the streams of the other participants from the server. Some of the weaker members may not be able to download a high quality stream. In that case, each user encodes his video in a format which is downloadable by the weakest peer.

The other approach to network heterogeneity is video coding at the server. The user encodes his video and sends it to the server. The server re-encodes the video in different bit rates for peers with different bandwidth conditions. This causes more complexity in the server implementation and increased latency in the video communication. As video conferences are bounded by a latency of around 300 ms for a good quality of experience, the server hardware must be powerful enough to encode the video in real-time and not affect the latency by a greater margin. More latency would impose restrictions on the maximum geographical distance between users in the video conference.

Alternatively, depending on the upload bandwidth available, video can be encoded into two different streams and sent to the server for the other users to download. As the server now does not have to perform network coding functions, the server does not have

to be very complex and the latency is reduced. Sending either a single or multiple streams of video to and from the server causes a huge bandwidth bottleneck and the organisation maintaining the server has to pay huge amounts of money for the bandwidth. Also in the case of a server failure, a secondary server which supports all the functionality of the primary server is needed to ensure that the system does not break down. Hence for a company with limited resources and engineering manpower, the client-server architecture would be difficult to sustain. Architecturally this option is not limited to any codec and a wide variety of options may be used. For session management, SIP servers are available which help maintain a session via a server. Such an architecture can be seen in figure 2.10.

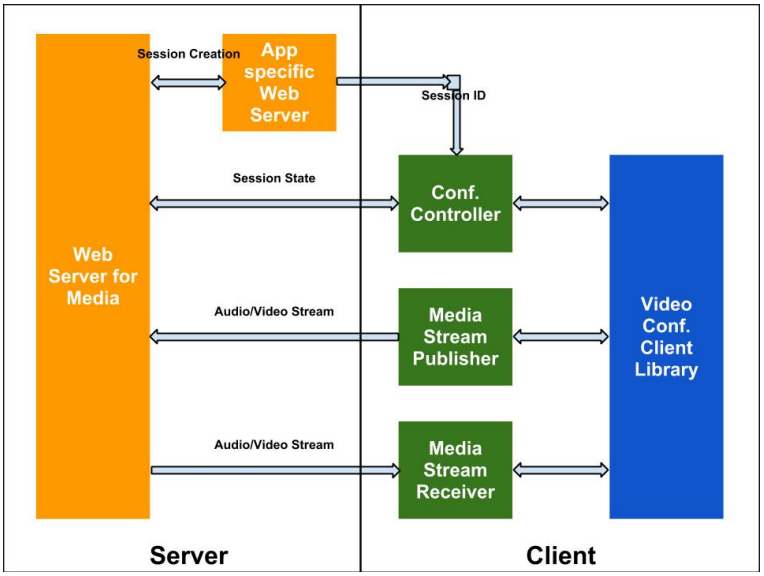


Figure 2.10: Client Server Architecture System Level Diagram

Google Hangout is a client-server based video conferencing solution. This architecture is feasible for Google given the high capacity of its already existing infrastructure. Google Hangouts used H.263 and H.264 codecs in the past (they have since switched to VP8) and are powered by Vidyo [70]. Following the design challenges which occurred in the development of Google Hangout, Google launched the standardization effort of WebRTC.

2.4.2 Peer-to-Peer

The other option at realising a multi-user video conferencing architecture is through a peer-to-peer system. The system can be viewed in figure 2.11 where each peer is connected to the other peer. This approach does not require the need of a powerful high bandwidth server and is relatively easier to maintain. A server is still needed for activities like maintaining the database of all active users, for signalling and for other activities like NAT traversal but as video distribution is not done through the server, the server does not need a high bandwidth internet connection.

However with more number of users joining the conference there is an upload band-

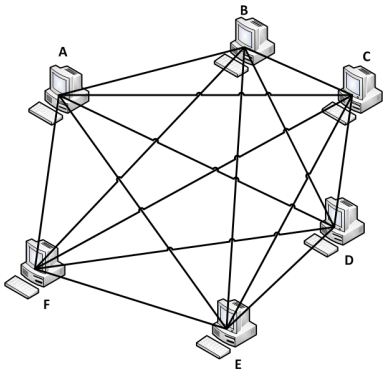


Figure 2.11: Architecture of a Peer-to-Peer Video Conferencing System

width bottleneck at each user. As upload bandwidth is less than the download bandwidth, this limits the size of the group considerably. Using this technology to realise a multi-user video conference only increases the upload and the download bandwidth demand. Thus users with slow connections cannot join a multi-user video conference. Video decoding by the weakest peer is still a concern and the source usually sends video of a quality which is decodable by every receiver. In this architecture, multi layered coding techniques can be used to have each user decide his choice of video so that no user has to content with lower video quality when he has the capacity to download and decode a video of higher quality.

Several video conferencing applications allow pair wise video conference. Skype uses this architecture for conferences up to 3 users. Many other applications have been built using this architecture as it is easy to maintain.

Peer-to-peer conferencing is inefficient, quality deteriorates very quickly as more users join. The incoming and outgoing bandwidth at the client is not the same and is even less in case of mobile devices. For multiple users, multiple encoding of video for different users takes up huge amounts of processing power as video encoding is computationally intensive. With suitable choice of technology, it can be done only once. But sending video to different users directly over the internet requires bandwidth and network processing power. Hence a media relay server seems to be the better solution in supporting multi-user chat. But maintaining a server is expensive and complex. The bandwidth at the server affects scaling capacities. In addition to that it transfers the processing time and complexity of maintaining a multi-user conference from each of the client to one central server resulting in the usage of significantly higher number of resources.

2.4.3 Peer-to-Peer with Peer Contribution towards video distribution

The problem of delivering high quality video in larger and larger groups of video conferences can only be solved by reducing the bandwidth footprint of a video conferencing system. Scalability of video chat takes a hit as the number of participants increases, both in the case of server based systems and peer-to-peer systems. Many approaches have been discussed to reduce the bandwidth footprint of a video conferencing system. One of them is introducing devices in the network which have access to a higher band-

width internet connection. These devices are called MCU (Multipoint control units). They receive video signals of various users and distributes them to all participants [88]. However the high cost and complexity of the system makes it suitable only for the use of large businesses who have the capability and resources to manage these devices.

However, in a different approach, a user who is one of the receivers of the video can be made to act as a MCU and be used to share video with other receivers. This approach is aimed at distributing the video signal efficiently by utilising the bandwidth of the users to distribute the video. This removes the bottleneck in the upload bandwidth as each user only uploads his video once and acts as a source in a live stream. The other users subscribe to all streams and realise a video conference. This approach has the benefit of scaling in the download bandwidth rather than the upload bandwidth. This approach is distributed and thus is robust as there is no single point for failure. As in a P2P system, each user is a stream generator who generates the stream of his live video to be subscribed by others, a user only uploads one stream of his video. The stream propagates via trees to users with each user acting as a node who consumes the video but also utilises his upload bandwidth by sharing the video with other user who wishes to consume it. This means the available upload bandwidth for dissipation increases with increase in number of peers subscribing to it. The data may dissipate in a tree like structure with nodes at the top of the tree contributing more in bandwidth as compared to the nodes at the bottom of the tree.

Systems like these are implemented using a robust P2P streaming protocol at the application layer. There are many systems which have been evaluated in literature as possible candidates for supporting a multi-user video conferencing system. They are Narada, NICE, ZIGZAG, End System Multicast, Overlay Multicast etc. The challenges in realising a video conference using these technologies are as follows:

- **Costly Framework** - These protocols have been basically developed as real-time video streaming systems. A streaming video system, although real-time may not have as hard real-time requirements as an interactive video conferencing system. A stream may start after an initialisation delay of several seconds followed by an end to end latency in excess of 300 ms. However a video conference should have minimal initialisation delay (less than 300 ms) and also the end to end delay should be limited to 300 ms in the worst case. In these conditions, a lot of engineering effort has to be put in to optimise the latency at various levels. Also to ensure robustness, lot of performance related optimisations may be required which will again take a lot of engineering effort. Although these systems promise scalability of greater orders of magnitude than traditional video conferencing systems, the cost-benefit ratio for these systems remains quite high.

- **Congestion Control** - Due to inherent structural difference between traditional video conferencing technologies and these, standard channel reservation or session maintenance strategies cannot be used. However congestion still affects these systems and a channel specific feedback cannot be obtained. On the other hand, many peers may have downloading capability more than they need and could upload multiple descriptions of videos. Hence the strategy employed here is that

every peer uploads multiple streams of a MDC video with every stream being every description. Peers download only the number of descriptions their bandwidth permits. Video packets may also be re coded according to the appropriate outgoing bandwidth at every node. This is called network coding but this leads to increased latency at every node of the tree which reduces the tree depth and severely limits the available bandwidth in the network.

- **Session Management** - Session initialisation and management is done using standard P2P handshakes querying for data availability and tracker address and at the application level, these handshakes may be too time consuming. This increases the overall complexity in session establishment and increases the initial connection time. Depending on how populated the network is with potential video distributing peers, the connection time may vary from a few milliseconds to several seconds. Such lack of reliability affects robustness making these systems less attractive for deployment scenarios.

- **Tree Creation** - The underlying structure which dissipated real-time video data amongst participants of the video conferencing system is the tree. The ideal tree would be characterised by nodes having the highest bandwidth being the higher branches whereas nodes having less bandwidth lower in the order. As each node will be a source of its own data dissipation tree, a group of participants would symbolize a mesh of different trees together. The depth of the tree of the peer with the weakest bandwidth would determine the maximum number of participants (which would be more than that of a traditional VC because of more available bandwidth in general). However the critical catch in this condition is the maintenance and repair of the tree in the case of additional peers joining or a peer leaving the conference. When newer peers join the conference, the peers with higher bandwidth should be placed nearer to the top while peers with low bandwidths should be placed near to the bottom. If a peer from the bottom of the tree leaves the conference, there is not much change in the available bandwidth but if a peer at the top of the tree decides to leave, the tree must be repaired in real-time to ensure that the conference quality of experience does not degrade due to loss of link or unsustainable video quality. This challenge requires a very complex and an adaptive algorithm and ensuring robustness of such algorithm in deployment conditions remains a challenge.

Skype audio calling software is famously known to use multi point peer-to-peer architecture using their own proprietary protocol (Kazaa) and video conferencing systems like Chatroulette use this architecture too.

Case study: Libswift - Peer-to-Peer Streaming Protocol

Libswift is a peer-to-peer streaming protocol developed at TU Delft. The Peer-to-Peer Streaming Protocol working group has adopted the Swift peer-to-peer streaming protocol (P2PSP) as a working group reference. This protocol is called Swift, for brevity. The basic difference between Swift and other P2P streaming solutions is that the other protocols are at the overlay level or at the application layer while Swift is at the transport layer. Applications like GetMiro, MyP2P, Coolstreaming [93] and Narada [65] have

shown that Real-Time Video can be streamed using P2P networking and Civanlar et al. [27] also demonstrated a video conferencing system developed using underlying P2P network. However as mentioned the main difference with the P2P streaming applications in them and in Swift is that unlike the other protocols, Swift is implemented at the transport layer.

Swift can be an ideal candidate for implementing a video conferencing system as it satisfied all the above requirements and in some cases it is even better than contemporary protocols available. Subsequent paragraphs discuss how Swift fares in dealing with challenges offered by a video conferencing system.

- **Framework** - Swift has been basically developed as video streaming protocol. Hence even though Swift may be capable to host a video conference, it has not been tested under hard real-time requirements. A lot of engineering effort has to be put in to optimise the latency at various levels. However, the handshake associated with starting a Swift download is less compared to other streaming systems like Bit Torrent and because of this inherent difference, Swift is easier to optimise for hard real-time requirements as compared to other P2P systems like Bit Torrent. However, the cost benefit ratio for Swift in terms of engineering effort would still remain quite high.

- **Congestion Control** - Swift uses LEDBAT (Low Extra Delay Background Transport) which aims to mimic a kind of congestion control which is TCP compliant. Congestion Information is available at the transport layer and hence the overhead induced in performing congestion control at the application layer is reduced.

- **Session Management** - As Swift was not originally designed for real-time interactive media communication, there is no concept of sessions associated with a Swift stream. Hence a lot of engineering effort would be expended in maintaining a session with one of the existing session management protocols. It would also be interesting to investigate whether the use of session management protocols is needed or not.

2.5 Quality of Service

Quality of Service (QoS) of any application is the guarantee that the application will meet certain basic performance requirements. Video conferencing applications are network based applications and hence Quality of Service can be defined as the performance measured from the network perspective at the packet level. However video conferencing is in many ways error tolerant and sometimes, little degradation in QoS may mean no degradation in the performance from a user's perspective. Hence careful measurement of QoS and the effect of change in QoS on the resulting performance from a user's perspective is needed to assess the robustness of the application. The QoS is subject to variations as network behaviour can be random and unpredictable at times and change in QoS is largely correlated to change in network behaviour.

The performance of the application from the user's perspective is just as important as the QoS. This performance is called Quality of Experience (QoE). QoE can be defined

as the QoS at the application layer. With regard to application performance, QoE and QoS are parametrically correlated with QoS being a performance measured at the network level and QoE being the performance measured from the user's perspective. Degradation in QoS can lead to degradation in QoE and hence by monitoring one of them, the other can be estimated to a fair degree. Measuring QoE accurately requires human experiments which are subjective, not scalable and are hence more likely introduce human error. It is comparatively more easier to measure QoS, identify and correlate QoS and QoE parameters, and then determine the QoE empirically.

If one wishes to build a high quality video conferencing application, the QoS should be carefully monitored to ensure that it does not drop below acceptable standards. In order to monitor the QoS, the network conditions should be monitored. As mentioned earlier, network conditions tend to be unpredictable and a tool is needed which can assert control over network characteristics and subject the video conferencing application through a variety of network conditions in order to asses its performance in the best case as well as the worst case.

Statistical data generated by common network monitoring tools may or may not be sufficient to provide an insight into the behaviour and robustness of VOIP and video conferencing applications. A dedicated tool needs to be used which can monitor the network conditions, control them for a brief period of time, collect information pertaining to the behaviour of the application and store this information in a database. This information can be used later for forensic analysis, recognising trending patterns, identifying trigger thresholds pertaining to various network conditions and for viewing the general real-time behaviour of the application. This information can also be used to provide recommendation for values of critical parameters leading to optimised and or better performance.

Various tools have been developed and they have been discussed in literature where an attempt has been made to characterise the behaviour of video conferencing applications. Cicco et al. [30] developed a framework to determine how Skype [75] adapts its video source rate as a response to changing bandwidth. Zhang et al. [92] developed a tool for profiling a Skype video call and also studied the variation in video source rate with variation in network conditions. Xu et al. [90] used a tool and analysed the topology, video adaptation and quality of experience of iChat [28], Google Hangouts [37] and Skype. Kuipers et. al used a tool for measurement study of multi party video conference in order to study their working, determine use of resources and bottlenecks. Schulzrinne and Baset developed a framework [20] for analysis of Skype VOIP protocol. Apart from these many other frameworks have also been discussed which generate data with varying degree of accuracy and need varying levels of manual intervention. Some of the frameworks need costly setups and a high level of manual intervention (and hence are time consuming) like the need to visually monitor a call, using a camera to record video quality etc. Some of them are designed to collect empirical data in a way which requires little or no manual intervention, sacrificing accuracy for speed. However all of them intend to test the effect of changing network conditions on the application performance.

2.5.1 Challenges to the QoS

Being a network application, the challenges to the QoS can be summarised as follows:

1. **Bandwidth** - Video source rate is dependent on video quality and when the available bandwidth is higher, higher quality video can be enjoyed by the user. Alternatively, the bandwidth can be used to accommodate more users in the call. A change in the bandwidth has to be compensated with change in video quality or change in the call group size.

2. **Delay** - Increased delay decreases the interactivity of the application. Variation in delay is known as jitter and studies have shown that users are more likelier to disconnect the call sooner if the jitter is higher [26].

3. **Loss** - Although multimedia applications like video conferencing are loss tolerant up to a certain degree, the user experience of such applications suffers if the loss increases beyond a limit.

4. **Heterogeneity** - In a two party or a multi party call, video stream decodable by one of the users may not be decodable by the other due to differences in hardware. Hence the video must be carefully customised so that it is decodable on weaker platforms with loss in video quality.

The tools discussed above attempt to quantify and measure application performance while changing the above mentioned parameters. Alternatively, the information can be also used to determine optimum operating conditions for a video conferencing library as follows:

1. The data can be used to determine what bandwidth is enough for a video conference. This information can be used to limit call sizes. Bandwidth usage is a function of video quality and thus this information can also determine the maximum quality which can be made available to the user in a video call.

2. It can determine the worst case network conditions (delay, packet loss etc) which a library can handle before the call becomes unsustainable.

3. A library may be configured beforehand to ensure that it meets some performance constraints (like maximum bandwidth, minimum video quality etc.). The data can be used to find out if the library overshoots the constraints placed on it.

4. Lastly, the framework can divulge information pertaining to topology (like location of the server or other peers) which can be utilised to optimise the back end infrastructure required by the library.

Moreover, data like delay, packet loss ratio and jitter can be used to model the congestion in the network and the resulting model can be used to design the optimum response of the library to the available congestion. Model based estimation techniques run the risk of being either too complex or oversimplified and therefore the data generated from these tools can be used to carefully calibrate the model.

Later chapters of this thesis discuss a problem statement of building a system which uses a video conferencing library as an engine. They draw requirements which are placed on the library, discuss existing libraries which meet a basic set of requirements. Later they discuss the design of a framework like the ones discussed above in order to determine whether a chosen library is able to deliver a minimum required QoE or not and present the findings of applying the framework to the chosen library.

Problem Description, System Requirements and System Design

3

This chapter discusses the initial steps towards the design of a video conferencing system for Presence Displays BV [9]. The product is aimed at the health care sector with special focus on connecting people with disabilities and psychosis through a virtual platform. The system is designed to be an application targeted on an Android platform (given the low cost) and is aimed at realising a virtual space where users meet for a multi-user video conference. This chapter lists the technical and design challenges posed by the design and the proposed development of the application. This chapter is organised as follows. Section 3.1 presents the unique aspect of this concept which differentiates it from other applications in the market. Section 3.2 discusses the use cases which need to be analysed in a user centric software design flow. Section 3.3 discusses the system requirements which are derived from the use cases and a system architecture and a modular design are presented in section 3.4. Section 3.5 concludes this chapter with future directions for product development based on the analysis presented in this chapter.

3.1 Concept

The video conferencing system in question which is being developed envisions the realisation of a virtual online space where users can meet fellow users using the same application and choose to engage in a video chat with them. In this virtual space, users will be represented by a thumbnail (or a bigger size window as per the screen resolution) of their video. The thumbnail sized video makes rendering of more and more users possible on a small screen (as tablet screens are smaller than laptop screens). The users will be able to engage in a group video chat with fellow users in the virtual space. Hence one of the unique aspects of this product is that it facilitates multi-user video conferencing with dynamic connections. As the unique aspect of the product is its user experience, a set of basic user requirements have to be satisfied at all times and hence a user centric design approach was selected.

According to the development model seen in figure 3.1 [12], the first part of the design is the analysis of user requirements and extracting the system requirements from that design.

3.2 User Requirements

The following sections analyse the important use cases to better understand the user requirements. Some of the user cases are not discussed in this document because of confidentiality purposes. The purpose of this section is to give the reader a common understanding about scenarios encountered in deployment.

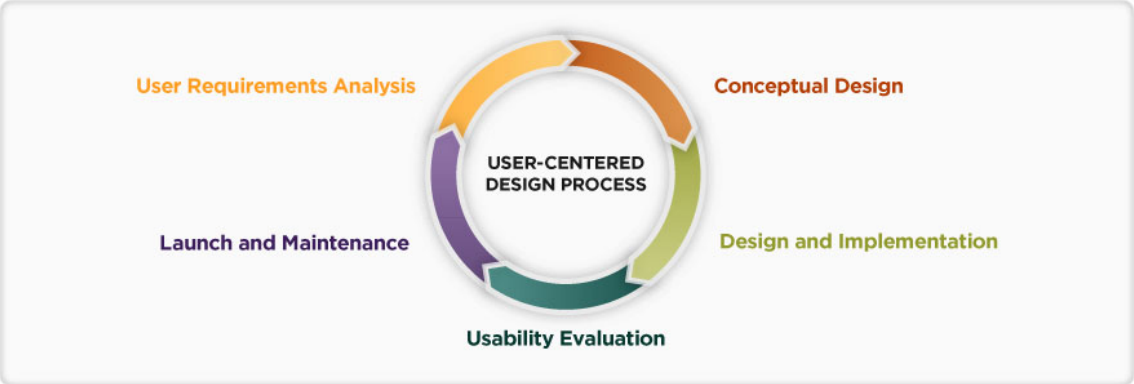


Figure 3.1: User Centric Software Design Stages

To better introduce the user scenarios, we define the following terms for brevity:

- **Server** - For centralised management, a server may be needed. This centralised conference management entity will be called the server.
- **Feed** - The combined audio/video generated by each participant in the conference will be called the *feed*.
- **Client** - There is software running on the tablet of every user. This user side software is referred to as the *client*.
- **Participant** - The average user who participates in the conference is referred to as the *participant*. We call them A, B, C and so on.

3.2.1 Login Scenario

In order to detect a new user who has just started a session and to ensure that only registered users are using the software, the video conferencing library in use will have to provide some type of user authentication like a login/logout service. This prevents the usage of the software by unauthorised users or users who are not meant to access the service at that particular point in time.

3.2.2 Group formation scenarios

As the application intends to let users make groups of more than two users, the video conferencing library must be capable of handling a multi-user video conference. To allow new users to dynamically join the conference (and to modify the user experience so that the user does not have to re-authenticate himself each time he joins a group), information on the login procedure of the video conferencing library is needed.

If more number of users subscribe to the chat, the video quality has to be lowered to accommodate the decrease in per user bandwidth. Once a stage is reached where an acceptable quality of video cannot be delivered, audio should take priority and the users should engage in only an audio conversation.

3.2.3 User shuts down/disconnects without warning Scenario

This scenario deals with the possibilities arising when the user does a non procedural shut down (disconnection without a warning). If the user does a non procedural shut down, the server disconnects the conference after a certain period of time and does a log out procedure for the client. The opposite of this scenario is the scenario where the server fails. If the server does not acknowledge an alive message or any message from the client for a certain specific amount of time, the client considers the server to be failed and stops all activity. It also notifies the user that the server has failed, so the user is not surprised with the sudden loss of connection. It probes the server with repeated periodic login requests to see if the server is back up again.

3.3 System Requirements

From the use cases, the requirements for system design have been formulated. These requirements will help us establish important criteria for the selection of software libraries/pre-built modules for realising the application. It will also give the designers a clear idea of what is expected of the software that they are building.

The requirements of the system can be divided into three parts:

1. Software system requirements
2. Video Conferencing Library requirements
3. Hardware requirements

3.3.1 Software System Requirements

The system requirements can be summarised as follows:

1. From a brief overview of the discussed scenarios, it is clear that the presence of a central server for administrative and conference management purposes is essential.
2. There should be a graphical user interface (GUI) which allows users to interact with the application and shows the video thumbnail of everyone on the screen.
3. There should be an audio/video renderer module present which displays feed from the other users in the system on the tablet screen of the user.
4. There should be an audio/video generator module which records audio/video of the user and generates media content which is streamed in real-time to the other participants in the conference.
5. The client and the server both should have a software module which performs the login and logout procedure. The server must be able to authenticate the validity of the users and do a login/logout procedure whenever requested by the user. These modules ensure that application is not misused and that only authenticated users are allowed access.

6. The user searches other users in the system in order to interact with them. Hence the client needs to know the other users if it wants to open a video conference connection with those clients. Hence, a requirement is low interaction times with a centralised connection server enabling these tasks to be done in real-time.

7. The client software should have a real-time video communications library which is equipped to handle multi-user video conferencing services. The multi-user aspect of the library is crucial as the application aims at supporting as many users as possible in a conference.

8. The connection establishment time of the library is a critical parameter for quality of experience. It has been observed that users tend to get dissatisfied with the call quality if the call establishment time increase beyond a limit [26].

9. The video quality needs to be changed depending on the congestion status in the group (which also depends on the group size). Thus the underlying library which implements the video conference needs to have varying or scalable video quality.

10. The application is expected to run on wireless networks in different geographical locations. In such networks, the connection quality depends on the geography and may vary over time. Hence the video conferencing library should have a strong and robust congestion control strategy.

11. As the application is designed for a tablet, it should maintain a strict power budget and CPU resources should be minimally used.

3.3.2 Video Conferencing Library Requirements

In this section, the necessary functional requirements for the video conferencing library are described.

- **A Server** is required to keep track of which group of users are in conference with each other. This server can also be a media relay server depending on the topology of the library.

- **Scalable Video Quality** - The video conferencing library in use should have scalable video quality so that video delivery is not impaired due to heterogeneity in bandwidth and hardware across users.

- **NAT Traversal** - The video conferencing library in use should have NAT traversal functionality like TURN (Traversal Using Relays around NAT) [55]. TURN is a protocol which enables clients behind a NAT or a firewall to connect with other peers via TCP or UDP. As media packets are sent over UDP, TURN or any other NAT traversal protocol is essential for establishing a connection with other peers for a video conference.

- **CPU Usage** - The CPU usage of the video conferencing library should be such that the running of other applications on the tablet should not be adversely affected.

- **Acoustic Echo Cancellation** - Echo cancellation is a problem encountered in real-time interactive audio applications and an AEC module must be present to eliminate the irritative effects of the echo.

3.3.3 Hardware Requirements

The requirements and the system design impose certain broad requirements on the choice of the implementation platform. The requirements can be summarized as follows:

- **Power Supply** - The tablet should be powered by a battery which can be charged with an adapter connected to the standard power supply (220 V).
- **Operating System** - The tablet should be running an Android OS.
- **Display** - For input/output, the tablet should have a standard multi touch display.
- **Speaker** - The tablet should have a stereo loudspeaker with a 3.5 mm jack for a standard headphone.
- **Camera** - The tablet should have a web cam and it should be capable of providing a resolution and frame rate comparable to that of other devices in the market.
- **Microphone** - The tablet should have a microphone for audio input with acceptable noise and crosstalk levels.
- **Internet** - The tablet should be capable of supporting a wireless LAN connection.
- **Hardware Acceleration** - The tablet device should have a GPU which can be used to accelerate encoding and decoding of the video.

Apart from the above mentioned requirements, there are some other practical requirements which the software needs to fulfil. Some of the requirements are as follows:

1. **Maturity** - A critical requirement while selecting a component is its maturity. A highly mature software already meets a large number of the general technological challenges in video conferencing and less engineering effort is required in making production code. Also a mature software also typically has a strong developer community which can be turned into for advice in case a fundamental challenge is encountered. A mature software component greatly reduces the resources used and the time to market. This is even more critical in the case of a company like Presence Displays which has limited resources owing to its size.
2. **Cost** - The cost of the components used in design should be as low as possible. Higher cost of development results in a higher price of the product and it takes a longer time to get financial returns from the product. This criterion is relevant for any company's business needs and becomes even more critical in the case of Presence Displays.

3. **Modular Structure** - Even though video conferencing is a technology which has been standardised at many levels, newer and newer standards are still being pushed [49] and newer and more efficient components are being developed. Integrating a newer and more efficient component in a product can be challenging and many newer challenges can be encountered which demand highly skilled know-how and engineering effort. To avoid these problems, the initial choice of implementation should be easily modifiable and newer components should be easily integrated into it.

4. **Intellectual Property Rights** - It is in the interest of Presence Displays that the copyright of the software developed remains within the company. It is essential that the knowledge acquired in solving the product specific challenges remains with the company. Hence the company should be able to white brand and not publicly disclose the software to the open source community.

Video conferencing is much researched technology which is 40 years old. Hence various modules which constitute a video conferencing stack have been optimised for specific use cases over the years. Anybody wishing to build a video conferencing application does not need to implement the complete infrastructure but can use already existing libraries and modules to design the system. The application centric modifications can be made to the library later which do not require as much engineering effort as compared to designing a video conferencing infrastructure itself. Hence the next step of design is to identify which existing components can be used for making this application. The requirements derived from this section give us a brief idea of the criteria that must be looked into before making the choice of which components to use in the product design.

3.4 System Design

From the previous sections, a clear idea has been obtained of all the requirements which must be satisfied by the system. This section tries to explain how the requirements were translated into system design. As the topology is the most important part of any system design, it is important to fix the topology first. The next sections discuss various topologies.

3.4.1 Topologies

For a video conferencing system, as already pointed out in earlier chapters, there are predominantly three types of architectures:

1. Client-Server
2. Peer-to-Peer
3. Peer-to-Peer with usage of peer bandwidth for video distribution

The challenges posed by each of the architectures are discussed in the following subsections.

Client-Server

In the client server architecture a robust server with a high bandwidth has to be implemented. This architecture lacks scalability and becomes difficult to maintain as the number of users increases beyond a certain number. Also the server is a single point of failure and hence a back up server must be in place to ensure seamless functionality. This architecture is flexible to heterogeneity in bandwidth and hardware across peers as the server can modulate video quality for each peer depending on its bandwidth and decoding capacity.

Peer-to-Peer

The peer-to-peer architecture is more cost effective from the perspective of the company as it company does not need to maintain a server for delivering audio/video data. This does not present a single point of failure and if there is a failure at any peer, only that peer is affected and the rest can continue their video conference. This architecture is less flexible to heterogeneity in bandwidth and hardware across peers as a single peer cannot send different qualities of video conference across different peers. Another drawback of this architecture that it is not group scalable. The upload bandwidth is less than the download bandwidth and hence uploading the same video to larger and larger number of peers does not scale in upload bandwidth. This problem can be seen in figure 3.2. Ensuring that each user gets the highest quality of experience available irrespective of hardware/bandwidth constraints is very difficult. Managing and tracking potentially thousands of users will increase the complexity in this architecture.

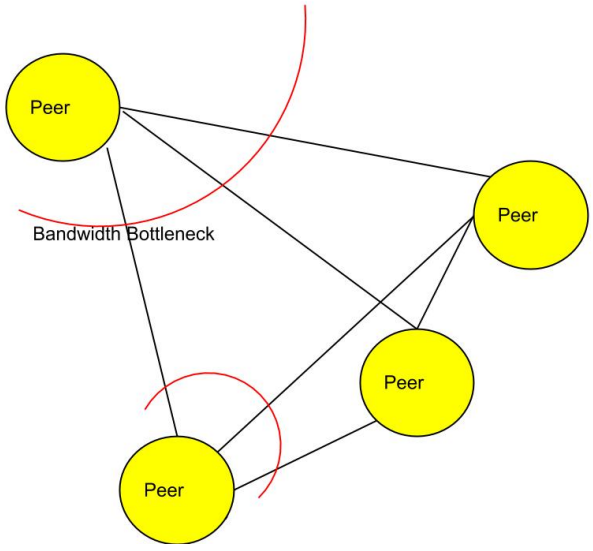


Figure 3.2: Peer-to-Peer VC

Peer-to-Peer with peer bandwidth usage for video distribution

This architecture is scalable in the download bandwidth and is hence more scalable than the one discussed previously. Here, each peer uploads his own video once and acts as a relay for video of some other peer. Each peer subscribes to streams originating from other peers it wishes to be in a video conference with. Each peer may receive the feed from a peer who has enough bandwidth to relay his as well as the source's feed. This approach makes it possible to utilise the complete bandwidth available in the system. Apart from that, as the application has number of users logged on to it at any given point of time, their bandwidth can be utilised for video distribution. This approach requires less back-end infrastructure and reduces infrastructure costs for the company. The drawbacks are that such systems are mostly tested for use in VOIP and very few have been tested for use in video conferencing systems. The engineering effort required to realise such a system is high. This architecture can be seen in figure 3.3.

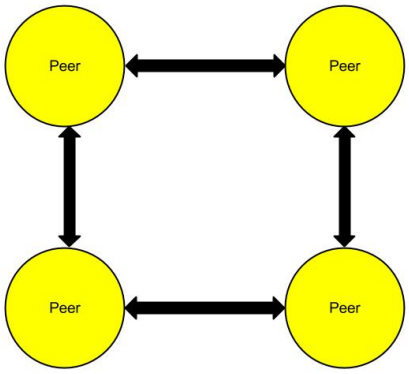


Figure 3.3: Each peer relays its own plus the video of another peer

The topology depends on the video conferencing library which is available for use in the product. Other parts of the system will then be designed keeping in mind the topology used by the library.

3.4.2 System Architecture

The above topologies are mainly for interactive media communication between users. However from the requirements generated by the application, one of the conclusions that can be drawn is that a server will be nevertheless needed for application specific purposes and whether there should be an additional server to relay media is a decision that will have to be taken depending on the availability of suitable video conferencing libraries. Hence from the analysis of various requirements a client-server based architecture was

proposed with the modular architecture for server shown in figure 3.4 and for the client in figure 3.5

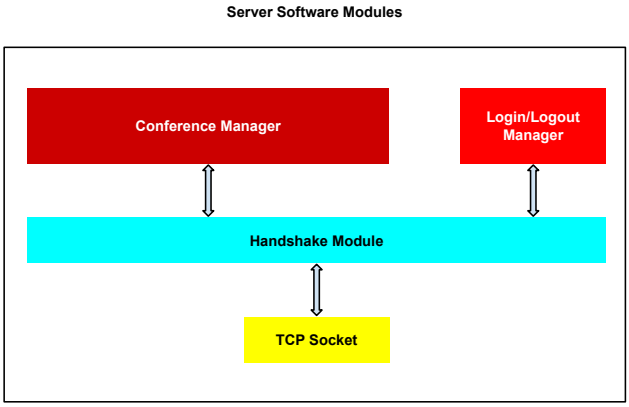


Figure 3.4: Server Modules

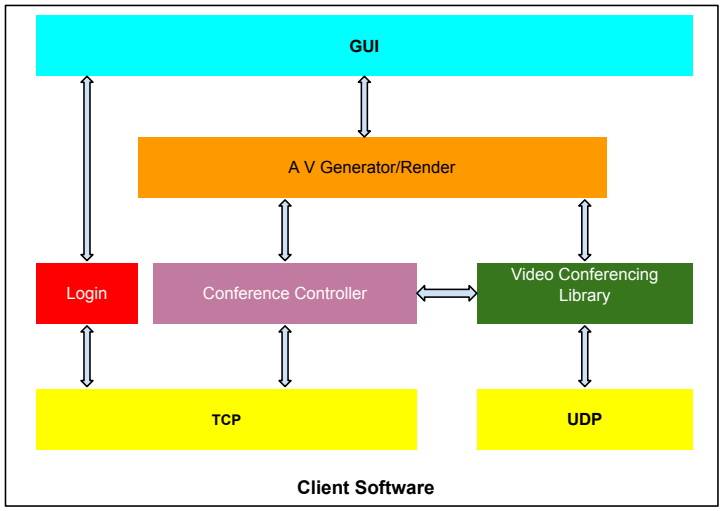


Figure 3.5: Client Modules

The software modules at the server can be described as follows:

1. **TCP Socket** - The server communicates to the client via TCP as a transport protocol. TCP is reliable and thus ensures that no packets are missed or lost.

2. **Handshake Module** - The server has a handshake module which tracks dead or a live activity from the client. If the client is requesting a login/logout procedure, or information about its peers necessary to make connection, then the server knows it is alive. If the client is not communicating with the server for these cases, as

may be the case when a long conference is in process, the server tracks the status of the client through periodic handshakes which are carried out by the handshake module.

3. **Conference Controller** - This module takes decisions on conference initiation, disconnection and quality of experience of the users. This module will be equipped to perform all conference management functions.

4. **Login/Logout Manager** - This module authenticates the user at initial sign in and prevents misuse of the application.

Similarly the software modules in the client software as shown in figure 3.5 are described as follows:

1. **GUI** - The GUI is the interaction point between the user and the software and is the layer where the unique user experience of the application is delivered.

2. **AV Generator/Renderer Module** - The audio video module records the audio/video and generates content for delivery to participants in the conference.

3. **Video Conferencing Library** - The content generated by the above module is sent through this module which is well equipped to handle multi-user video conference.

4. **UDP Socket** - User Datagram Protocol is used as the underlying protocol for media delivery as TCP is not well suited for inelastic media transport.

5. **Conference Controller** - This module communicates with the controller module at the server and depending on the requests made by the user or decisions made by the server, initiates, disconnects or takes decisions pertaining to the quality of experience in the conference.

6. **Login/Logout Manager** - This module is responsible for the user authentication initial sign in.

7. **TCP Socket** - The modules need quality of assurance at the packet level to ensure that none of the elastic data is lost. Hence The modules communicate with the server through a TCP socket.

Figure 3.6 shows the interaction between the client and the server at the modular level for the login/logout process. The login module at the client communicates with the login module at the server through a TCP socket.

The complete interaction process can be clearly understood with the help of the following user interaction and sequence diagrams. The user interacts with the client software through a graphical user interface as seen in Figure 3.7. Then the client does a login procedure with the server and authenticates the client. As the video conference starts via the video conferencing library, the video is rendered through the GUI for the user via the renderer module. The conference manager at the client continuously

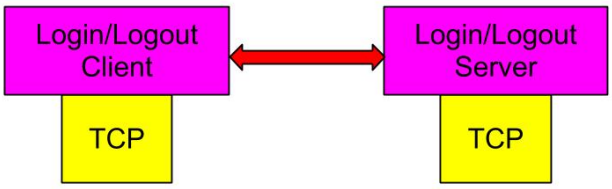


Figure 3.6: Interaction between login modules

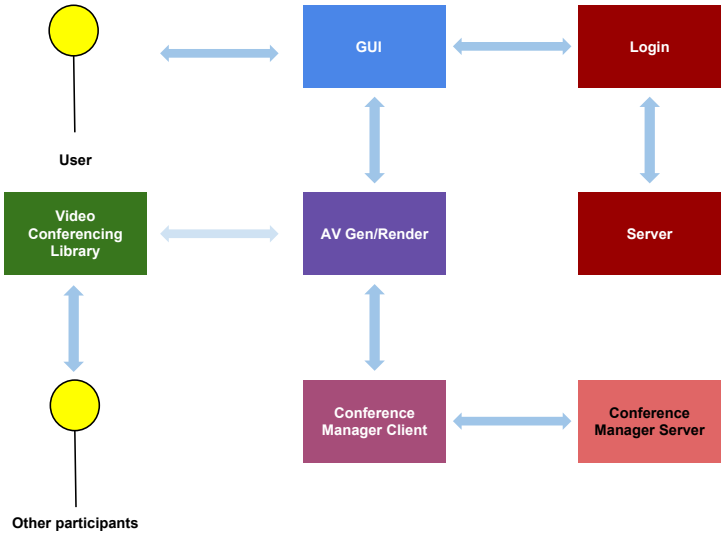


Figure 3.7: User interaction with the software

interacts with it's counterpart at the server. It also takes user preferences from the GUI and the changes in conference structure/experience etc. can be seen on the GUI.

The sequence diagram seen in figure 3.8 better explains the chronological order in which interactions take place between two users, A and B, leading to connection being established between them. User A initiates the login process through a login module present at his end. The login module requests authentication and waits for validation from the server. The conference manager module of User A then requests connection details of User B to the server so that a video conference between both the users can be initiated. The server then provides the conference managers of both Users A and B with the connection details of each other. These processes are invisible to the user and require no input from the user. As soon as the connection details are exchanged, the

conference manager module of User A establishes connection with User B notifying that a video conference is about to take place.

Once a connection has been established, a video conference is initiated by the video conferencing module present with both the users. As the conference is established, the the video of User B is rendered on the GUI of User A and vice versa.

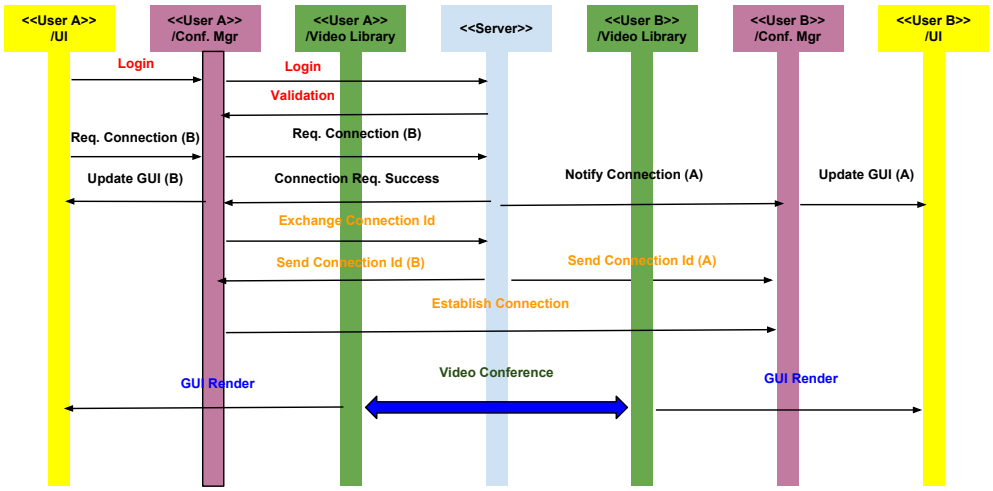


Figure 3.8: User interaction with the software

3.4.3 Server's function in QoE management

The application is committed to offering a minimum Quality of Experience (QoE) to the users taking part in the group. Thus if the bandwidth is not enough to provide an acceptable quality of experience, the application prohibits more users from joining the conversation. Usually QoE is the prerogative of the video conferencing library and measures taken to maintain a decent QoE such as encoding video to use less bandwidth, putting a cap on number of users for congestion control are implemented by the video conferencing library itself. As our application is built on top of the video conferencing library, the application server can also be used to supplement the congestion control features of the library.

All users periodically transmit information like alive handshakes and position co ordinates with the server. This channel which has already been established can be used to transmit sender's and receiver's reports about congestion and QoE. This approach can be used as a supplement to the congestion control strategy of the video conferencing library and not a replacement as robust congestion control is difficult to implement. Each user can relay important parameters like RTT (Round Trip Time), current video resolution, frame rate at their end to the server. The server can then pass on this information to other users or to the media relay server who then take appropriate steps to reduce congestion. Because of this approach the server has a constantly updated knowledge about QoE parameters which can be intelligently used to deliver varying

levels of service to the users.

Also in the future, the server can be used to realise the following functions:

- Additional software modules can be installed at the server to keep a track of any user's preferences.
- The server can be used to support server based applications for users in the future.
- The server can be used to send software updates to users if and when the need arises.

The following figure 3.9 shows the deployment scenario of the application in brief where it can be seen that different clients are connected to one centralised application server.

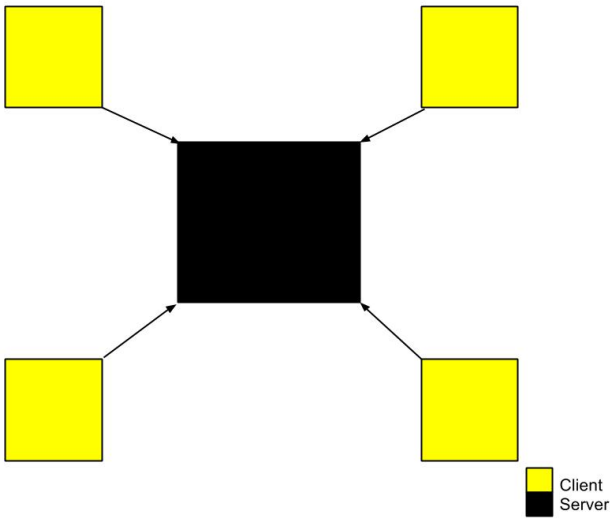


Figure 3.9: Deployment Scenario with a Centralised Server

3.4.4 Hardware Device

Google Nexus 10 was chosen as the tablet device for implementation as it fulfils all the necessary requirements to support a multi-user video conference. The features of the tablet pertaining to our requirements can be seen in table 3.1. The tablet runs Android OS and is thus cheaper as compared to other tablets running different operating systems. It has a large display of around 10 inches and thus rendering space is larger as compared to other smaller tablets and mobile devices. Video capture is available at 1080p and 30 FPS which suggests that a true HD video conference can be possible on this tablet. Nexus 10 also has a powerful GPU accelerator which aids suitably in video rendering.

Table 3.1: Google Nexus 10

Requirements	Google Nexus 10
Power Supply	Li-Po 9000 mAh battery, 9 hours Talk Time USB charger with a 220 V adapter.
Operating System	Android v4.3
Display	Multitouch, Super PLS capacitive touch screen, 2560 x 1600 pixels, 10.1 inches display (299 ppi)
Speaker	Stereo speakers with a 3.5 mm audio jack
Camera	5 MP, 25921936 pixels, Video capture @ 1080p (30 fps)
Microphone	Noise levels of -82.3dB Crosstalk levels of -81.4dB.
Internet	WLAN Wi-fi 802.11
Hardware Acceleration	Mali-T604 GPU accelerator

3.5 Conclusion

After the various requirements and functionality of different modules has been finalised, the next step is the identification of available video conferencing libraries in both the open source and closed source domain which will be used to implement the video conferencing software. The requirements extracted and discussed in this chapter will heavily decide which libraries are deemed fit for implementation. These libraries will then be tested for various parameters like robustness and maturity and a qualitative and quantitative performance analysis will be done to choose one final library for implementation.

Overview of Video Conferencing Libraries

4

4.1 Introduction

This chapter discusses different video conferencing libraries which can be used for implementation in the development of the application for Presence Displays. The system requirements were proposed after analysing the user scenarios and based on these requirements a system design was proposed for the video conferencing system. For the implementation of the system, the next step is the identification of software modules in both the open and closed source domain which can be used. The modules at the server are highly application centric and need to be designed and developed from scratch. However the other part of the system, which is the video conferencing infrastructure need not be developed from scratch as many libraries are available both in the open source and closed source domain. Hence the next logical step in the development of the application is identifying which video conferencing library suits best for the given set of conditions and proceeding with that library for implementation. In this chapter the available libraries were surveyed and some relevant libraries were found which were chosen for further development. Section 4.2 discusses the important criteria which were considered while narrowing down the list of libraries for development. In section 4.3 the available libraries which can be considered for development are listed and in section 4.4 the choice of libraries is narrowed down in a detailed discussion.

4.2 Criteria for Evaluation

Chapter 3 discussed the requirements needed for video conferencing software design. However not all the video conferencing libraries in the open and closed source domain match all the requirements. Hence some criteria is needed for short listing the libraries. The chosen library should have a good quality of experience. However testing all the libraries for quality of experience is time consuming. Therefore the criteria are to be divided into two sections. In the first section, the libraries are scanned for non functional but highly critical requirements like availability of code etc. The second criteria which will be tested is the quality of experience (QoE) itself. The short listed libraries of the first section are then subjected to an extensive qualitative and quantitative analysis for quality of experience and a library which suits all the requirements is selected.

The highly critical requirements which can be summarized as pre-selection criteria for the library are described in the following sections:

4.2.1 Quality of Experience

Quality of Experience (QoE) of a video conferencing library is the general user experience each library provides. Although highly subjective, it can be measured using various qualitative and quantitative parameters like frame rates, variation in frame rates, frame resolutions, variation in frame resolution, audio quality, communication delay, variation in delay, etc. The criterion that the library should have decent QoE has a high priority.

4.2.2 Maturity

Presence Displays is a small company with limited resources and engineering man power. Hence the primary requirement for selecting a library is that it is highly mature so that less engineering effort is expended in developing a product. A mature library already solves various basic technological challenges and thus reduces the time to market as a lot of time is saved in making product oriented optimisations into it. This criterion has a high priority.

4.2.3 Robustness

Robustness to unknown network conditions is an important criteria which is required for the library. A production version which handles unexpected network conditions is needed as the company cannot make fundamental changes to proprietary software in case changes are required for ensuring robustness. Although the actual robustness of the library under varying network conditions will be tested later to evaluate the performance, a basic guarantee of robustness through discussions in developer community and users will be much appreciated. This criterion has the highest priority.

4.2.4 Technical Support

There should be an active developer community so that challenges could be solved ensuring a fast time to market development cycle. An active developer community is a big help to the engineers designing and maintaining the software and makes it easier to debug and add future functionality. Considering the options or the lack of thereof, the priority for this criterion is low.

4.2.5 Android Compatibility

The application to be developed is intended for use on Android and hence the video conferencing library should be able to compile and work on an Android platform. Another option could be porting an already existing library to Android, but that would require a lot of engineering effort and many newer challenges would have to be solved in guaranteeing robustness in deployment conditions. If an appropriate solution is not found, a library which works on Linux systems can be ported to Android with extensive modifications. This criterion has a high priority.

4.2.6 Multi-User Video Conferencing

The library should be able to allow a multi-user video conference. The session management and bandwidth usage scenarios differ in case of a two-way conference and a multi-user video conference. Thus modifying a library which supports only a two-way video conference into a four-way video conference would require a lot of engineering work in ensuring robustness for different conditions. In order to avoid the challenges of porting a two-way video conferencing library to a four-way video conference, this criterion also has a high priority.

4.2.7 Availability of Source Code/API

The software library should be open to developers so that any application oriented customisations can be made to it. If the library does not have a developer API, we cannot make our modifications to it. This pre-selection criterion is a highly critical as a library cannot be used if the developer API is not available.

4.2.8 Price

The cost of obtaining the library for development should be relatively low and affordable. This criterion also has a high priority as a high cost solution increases the development cost and the time to fetch returns on investment. Also as the application is targeted towards the health care sector and people with disabilities, the cost of the application cannot be high as it would make it unaffordable.

4.2.9 Intellectual Property

Any modifications done to the library will be the result of engineering effort, time and resources expended by the company. Thus the company should enjoy the sole right over these modifications. There are commonly three types of licences:

1. **GNU Public Licence** - If the library carries a GNU Public Licence (GPL), the entire application which uses the library will have to be made open source. This means that the uniqueness in the application will be out in the open source domain and the company will lose any strategic advantage it might have had due to the novelty of the concept.

2. **Lesser GNU Public Licence** - If the library carries a Lesser GNU Public Licence (LGPL), any modifications made to the library will have to be made open source. However other modules which do not need to be compiled with the library can still be kept secret to avoid breach of company confidentiality.

3. **Berkely Software Distribution Licence** - Berkely Software Distribution (BSD) Licence is a more company friendly licence which allows the company to own the copyright and not release the modifications of the source code to the public.

Some libraries with a developer API come with clauses like the software needs to be evaluated by the developers of the library and that the developers may prohibit the use

of the library for that application in the future. These clauses cause a lot of uncertainty for future use of the application and application development itself. Therefore these intellectual property issues need to be carefully evaluated before choosing a library. Finding a library with an appropriate licence has a medium priority.

4.2.10 Support for other functionalities

The application expects to support other functionalities in the future like file sharing and chat. If the video library can support these applications, future development on the application will be easier. However, this criterion has low priority and if a library is found which does not support this criterion, it is still usable.

4.2.11 Summary

All the criteria priority and the parameters used in their evaluation are summarised in table 4.1. A 'Good' evaluation parameter stands for acceptable levels of QoE. An 'OK' parameter means acceptable level and 'Bad' means an unacceptable level. However as it will be shown, QoE is a highly subjective parameter requiring extensive analysis which is discussed in later chapters. If the source code or the API of a library is available, the parameter is 'Yes' otherwise 'No'. In case of favourable licensing issues, the library is given a 'Feasible' rating or otherwise a 'Not Feasible' rating is given to it. Likewise other parameters are used to determine where exactly a library ranks in order of preference when it comes to pre-selection criteria. As criteria like robustness and QoE cannot be determined at the pre-selection time, only the other criteria are used to select a library. The libraries would then be subjected to a framework discussed in later chapters in order to evaluate robustness and QoE.

Table 4.1: Priority of Criteria and Evaluation Parameters

Criteria	Priority	Evaluation Parameters
Quality of Experience	High	Good, OK, Bad
Availability	High	Yes, No
Maturity	High	High, Low
Robustness	High	High, Low
Android Compatibility	High	Yes, No
Multi-User	High	Yes, No
Price	Medium	Acceptable, Not Acceptable
Technical Support	Medium	Available, Partly, None
Intellectual Property Issues	Low	Feasible, Not Feasible
Other activities	Low	Possible, Not Possible

4.3 Overview of Video Conferencing Libraries/Applications

This section presents a review of most common video conferencing libraries available in the public and private domain which may be available for use. The domain space for VOIP (Voice Over Internet Protocol) applications reveal the existence of many VOIP libraries and applications. The search for these libraries was narrowed down to the applications that support video chat as the initial criteria to reveal an extensive list. The list is as follows:

1. **AOL** - AOL (America On Line) Instant Messenger is a video chat service which is made for Macintosh OS and Windows. It is also capable of file transfer and placing voice calls from PC to telephones. It is closed source and no API is available for developers to develop and build their own application using the library [16].

2. **Bria** - Bria is a program developed by CounterPath Corporation which is a video conferencing client. It is available for Windows, Macintosh. A Linux version is also available but only voice calling features are implemented. It is closed source and no developer API is available for building applications [29].

3. **Ekiga** - Ekiga is a free software available under the GNU public licence and is available for Windows and Linux platforms. The source code is available and hence newer modifications can be added onto it. It also has support for making phone calls to telephones via PC [31].

4. **Eyeball** - Eyeball chat is a free application for windows platform and along with video, it also has voice calling feature in it. Eyeball measured a nine in the top ten video conferencing softwares according to this review [1]. However the software is not open source and there is no developer API available for development [33].

5. **Eyebeam** - Eyebeam is also a program developed by CounterPath corporation and is a proprietary video conferencing application developed for Windows. It supports other features like voice calls. But a developer API has not been released for this library [29].

6. **Google Talk** - Google Talk is a voice, video and chat client available for Linux, Windows, Macintosh OSX and Android, and is available as a plugin. As the name suggests, it is developed by Google and it is closed source. Developers have access to an API allowing them to integrate hangouts within their application but the API does not make it possible to have a customisable independent client [37].

7. **iCall** - iCall is a voice and video conferencing client available for Linux, Mac, iOS, Windows as well as Android. It is a free software for download but a developer API is not available for developer services. It however supports other applications like file transfer and instant messages [17].

8. **Jitsi** - Jitsi is a free software for Linux, Mac and Windows operating systems. As it carries with it a Lesser GNU Public Licence (LGPL), its source code is available

and any modifications to the code will have to be made public. It supports features like text messaging [46].

9. **KPhone** - KPhone is a free software developed for Linux available under the GNU public Licence. Its source code is available for development as it is a free software. It has support for voice chat, instant messaging and IPv6 [48].

10. **Linphone** - Linphone is a video and instant messaging client available for all operating systems such as Windows, Linux, Android, Mac, iOS and even Blackberry. This software is a freeware and is available under the GNU Public Licence. Hence any changes to the source code will have to be made public under the GPL as well. However it is possible to get a commercial licence from the company. This library supports a lot of audio as well as video codecs and also supports high definition video provided the CPU is capable enough [52].

11. **MicroSIP** - MicroSIP is a free software for video conferencing, voice chat and instant messaging. It is available freely under the GNU Public Licence but is available only for Windows. As the name suggests, it uses the Session Initiation Protocol (SIP) for session management [58].

12. **Mirial Softphone** - Mirial Softphone is a proprietary software available for Windows and Mac OS which supports a High Definition and a full multi party video conference. However, it is closed and no developer API is available for customising applications [51].

13. **OOVOO** - OOVOO is a free software available for Mac OS, iOS, Windows and Android and supports various features like calling from PC to phone, instant messaging, and a full video conference with up to 12 users. This software is popular among users and came in at number 4 in the top ten list of video conferencing softwares [1]. A free developer API was also available for the developer community until 2007. This has however been discontinued. An API can be obtained by paying a registration fee and the choice of this library would depend on the cost benefit analysis in terms of robustness and scalability [60].

14. **QuteCom** - QuteCom is a free software available under the GNU Public Licence and hence the source code is available to the developer community. The application is available for Windows, Mac and Linux. It supports other features like instant messaging and can be integrated with other chat clients [68].

15. **Revation Communicator** - Revation Communicator is an application available for Windows and is proprietary in nature. It supports full multi party video conferencing, voice conferencing, instant messages, file transfer and even desktop sharing. However the source code is not available and neither is a developer API [69].

16. **Sight Speed** - Sight Speed is a free ware developed for Mac OS and Windows. It supports features like multi party calling and voice communication. But a developer API is not available. This application was rated at number 7 in the top ten reviews for video conferencing software [1].

17. **Skype** - Skype is a Microsoft owned free software available for all platforms like Linux, Mac, Windows, Android and iOS. It supports features like instant messaging file transfer, audio chat, and multi party audio chat. A multi party video chat option is available but is not free. Skype is one of the most popular software packages available for video chat and its robustness is well known to users all across the world. As it is a closed proprietary software, the source code is not available. However a developer API has been made available for the developer community for customising the testing. This developer API along with a testing framework is also available for Android. Hence Skype would be a good choice for implementing a multi-user application. However Skype was rated at 11 amongst popular video conferencing software [1] [59].

18. **TokBox** - TokBox is an application which can be used through browsers and is also available for Mac, Windows and Android. It features at number 6 in the top ten reviews for video conferencing software in 2013 [1]. A developer API is available for TokBox and can be used to develop applications for Android platform [79].

19. **X-Lite** - X-Lite is a software developed for Mac, Linux, and Windows. A developer API is not available and the support for Linux applications has been discontinued since 2012 [29].

20. **Yahoo Messenger** - Yahoo Messenger is an application for chat, voice and video conferencing available on all available platforms like Mac, Windows, Linux and Android. It is free to download but is closed source and hence the source code is not available for modification. However a developer API is available for instant messaging and file transfer and not for video conferencing. Yahoo features at number 2 in the list of top ten video conferencing software [1] [91].

21. **Acrobits** - Acrobits is a video conferencing application developed by Acrobits which is a leading provider of VOIP solutions. It is developed for mobile phones and hence would be better suited for the application in question. However a developer API is not available and neither is the source code available [13].

22. **Farstream** - Farstream is an open source project aimed at providing a framework which supports all audio and video conferencing protocols [5]. This library has good documentation support and as it carries a Lesser GNU Public Licence with it, the source code is available [5].

23. **OPAL Library** - Open Phone Abstraction Libary or OPAL is an open source library for VOIP and multi-user audio/video conferencing. The library is open source and hence the source code is freely available. The VOIP client Ekiga discussed before is based on this library. However, this library is too experimental and hence is not mature enough for production deployment [61].

24. **Vidyo** - Vidyo is a video conferencing application available for all major platforms including Windows, Linux and Android. An API of Vidyo is available for a developer fee [82].

25. **Auralink** - Auralink is an application developed for Windows and Mac OS. It supports a multi point chat and a javascript based developer API is available for integration into browsers. But an Android based API is not available for development. An android version is available but not much documentation is available about it [18].

26. **Radvision** - Radvision is an Avaya company which develops proprietary video conferencing applications for business needs. A developer API is not available directly but a sales representative would have to be contacted in order to negotiate the availability of an API at a certain fee.[19].

27. **AndBang** - AndBang is a WebRTC based application which is available for all operating systems including Android. There is also a developer API available, but the price is too prohibitive to use. There are also similar API's using WebRTC available via applications like Plivo and AT&T. However they need to be purchased and the cost benefit analysis must be done before making a decision to continue with them. WebRTC is an open source framework developed by Google which aims at integrating video conferencing functionality into browsers [10] [14].

28. **WebRTC** - WebRTC itself can be used to develop a video conferencing application and its source code is available freely. We have even compiled a WebRTC demonstration on Android working on the tablet. However multi party functionality has not been built into WebRTC yet and developing it will require a lot of engineering effort and time. However, WebRTC is a very promising candidate for implementation as a very strong developer community is active and newer and newer functionality is being added to WebRTC day by day [10].

29. **SIP on Android** - Using Session Initiation Protocol and the Android API, a SIP based VOIP application can be created. As the SIP protocol can also be used for signalling in video conferencing sessions, a video conferencing application can be developed in Android. However thus quite a large extent of the video library would have to developed and providing robustness would require a great amount of engineering effort [41].

30. **TringMe** - Tring Me is an application for video conferencing through browser and has a PHP based API for development. The API supports Windows mobile but support for Android is not available [80].

31. **PJSIP** - PJSIP is a free open source library developed in C for implementing standard real-time communications technology like SIP, SDP, RTP, STUN, TURN and ICE [66]. Multi-user video conferencing might be possible in PJSIP and Android support is also available but there is no information about the stability and maturity of this application. A proprietary API may be made obtained by contacting the company [66].

4.4 Chosen Video Libraries for Implementation

Each of the above libraries and some others were evaluated on the pre-selection criteria to narrow down the list of suitable libraries. Due to the unique constraints placed by licensing issues, availability for Android and maturity, only one library was available in the end. This library is called **Library 1** from now on in this thesis.

Library 1 - A developer API (for Android) of the library was purchased by Presence Displays on parameters of cost based analysis and availability of technical support. The API of the library was found to be highly mature and the functionality could be changed (partly) after playing with the API. Due to presence of such a robust API, low development times could be anticipated. The library supported full multi-user video conferencing and hence was well suited for our needs. It has internal performance measurement tools which make performance analysis easier for application developers.

Technical support is made available by the vendors and extensive online documentation is available on the software. This provided pointers to the application developers to tweak the API in order to match the needs of the application thereby ensuring efficient integration of the video conferencing library in their application. Furthermore Library 1 allowed integration of proprietary software with the library without the obligation of sharing it with the open source community or even the vendors of the library themselves. Table 4.2 summarises the evaluation of Library 1 using the pre-selection parameters. None of the important criteria scored a 'Bad' or a 'Non Acceptable' justifying the choice of Library 1.

Table 4.2: Evaluation of pre-selection parameters for Library 1

Criteria	Library 1
Availability of Code/API	Yes
Maturity	High
Android Compatibility	Yes
Multi-User	Yes
Price	Acceptable
Technical Support	Available
Intellectual Property Issues	Feasible
Other activities	Possible

Before selection of Library 1 for application implementation, a robustness analysis and a QoE evaluation had to be done. The next section introduces the concept of QoE in detail and describes what does QoE consist of and what metrics affect it.

4.5 Criteria for Quality of Experience

The most important criterion which needs to be evaluated is the Quality of Experience. Quality of Experience is a combined output of various subjective and objective parameters. These parameters are mostly related to video quality, audio-video synchronisation,

quality of interaction and general experience with the usage of the application. This section discusses parameters related to QoE, as follows:

1. **Video Quality** - Video quality is a subjective parameter which would vary on users perception of what is good video quality and bad video quality. However to evaluate video quality scientifically, we need an objective parameter to compare video quality. This parameter is the Batch Video Quality Metric (bVQM)[2] or MSU video quality measurement metric (MSUVQM)[8]. A score is obtained by comparing the video quality of the received video stream with the video stream that is generated at the source. In the case of bVQM, the bVQM score is usually between 0 and 1, with 1 being a score for very bad video quality and 0 for good video quality. However evaluation using this format takes a lot of standardisation and hence video is gauged to be good, tolerable or bad by observation.

2. **Audio Video Synchronisation** - Many users experience lack of synchronisation in audio and video in a video conference and that leads to general degradation in the quality of experience. The International Telecommunications Union suggests that a lag of about *+45 ms* to *-125 ms* is indistinguishable and a lag of *+90 ms* to *-185 ms* is acceptable [4]. However any delay after causes great degradation in quality of experience. These audio-video lags are caused by frame losses. Measuring audio-video synchronisation also takes a lot of time if it has to be done accurately. Hence the audio video synchronisation is quantified to be either good tolerable or bad.

3. **Audio Quality** - Due to bandwidth constraints, hardware constraints or reasons like privacy, the library should substitute the video conference by an audio conference. Having audio conference capability is of the highest priority and audio is gauged to be either good or bad.

4. **Delay** - The main different between a video conference and video on demand despite both being real-time in nature is that video conference is interactive in nature. This means that the quality of experience is closely related to quality of interaction. The interactive nature is dependent on the communication delay between the source and the receiver. It is generally known that the largest value of acceptable delay in a video conference is *300 ms* and if the delay exceeds that, the participants have to resort to tactics like saying 'over' at the end of each sentence which takes the real-time interactiveness out of the application. Communication delay can be quantitatively assessed in terms of milliseconds and hence no qualitative assessment parameter is decided.

5. **Dynamic video rate control** - Variable data rate is by far the only reliable solution to congestion control and considering that our application is going to run on a wireless channel and in larger group sizes, congestion control has a higher priority over maintaining a high video quality throughout the call. Hence the chosen library should have variable data rate (can be realised via change in frame rate, video resolution, quantization parameters). However considering that almost all video conferencing libraries have variable data rate to combat congestion, the

deciding factor becomes which libraries has a robust strategy which allows for smooth degradation in video quality. Assessment of this parameter is difficult as variable data rate affects video quality but improves experience by ensuring that there are no disconnections. Hence this is gauged in three categories:

(a) **Not present** - In this case, the library has zero rate variability and in the face of congestion, the quality of experience reduces drastically and in some cases, the call may be dropped.

(b) **Coarsely tuned** - In this case, rate variability is present but is coarsely tuned. This means that if congestion increases even slightly than a particular threshold, the data rate will be reduced considerably. This is less efficient in use of bandwidth and resources but less complex in computation on the part of the library.

(c) **Finely tuned** - In this case, the data rate is finely adjusted with changes in congestion and this ensures that the bandwidth and resources are properly utilised and the user gets the best quality of video possible at all times.

6. **Power Consumption** - As the application is meant for an embedded computing platform (a tablet) the power footprint of this library has to be low. A library which consumes more power threatens to drain the battery of the tablet and makes it unattractive for use on tablets. If a user at the end of his session finds that the battery is drained, he will not be happy with the general performance of the application - which will go further as to reduce the quality of experience. A numerical cap cannot be put over the power consumption for evaluating each library. However if power consumption is noticeably faster (battery gets drained in a short period of time), this raises a red flag suggesting that detailed analysis of the power profile of the library is needed.

7. **CPU Usage** - Higher CPU usage causes to more consumption in power and also may reduce the performance of applications running in the background. If that happens, the user experience with the application degrades to a great extent. Hence care must be taken that the application has low CPU footprint. If the performance of other applications noticeably suffers due to high CPU usage by the library, this signifies undesirably high CPU usage.

4.5.1 Metrics affecting QoE

Superficial evaluation of the above mentioned criteria is not enough for choosing a library for implementation. A more detailed analysis should be done to find out which differences in these libraries cause noticeable difference in QoE. The following metrics affect the quality of experience due to various parameters:

1. **Signalling Overhead** - Signalling data which is required for session management and handshaking is sent through TCP. The initial connection delay depends on how heavy the signalling is. This signalling is different for different protocols like SIP (Session Initiation Protocol), H.323 or XMPP (Extensible Messaging and Presence

Protocol) and an analysis of the signalling overhead can give us an idea about the connection establishment time as well as the bandwidth expended in maintaining the connection. A high signalling overhead requires high bandwidth. Although this does not majorly affect the video as the bandwidth used for signalling is still much higher as compared to video bandwidth.

2. **Traffic** - Real-Time traffic is inelastic and is sent through UDP. By monitoring the UDP traffic, we can find out which library generates on an average more traffic. The library generating more traffic more likely delivers better video quality but is also susceptible to congestion and less likely to be scalable for a multi-user video conference. A high traffic rate denotes efficient usage of bandwidth and is a sign of high quality video. However the call is then more susceptible to bandwidth changes and the conference becomes less scalable. A low source rate denotes lower quality of video, inefficient usage of bandwidth but higher robustness in face of varying network quality and more scalability (as more users can share the bandwidth). As source rates exceeds the available bandwidth, this causes increase in RTT and thus increases end to end communication delay.

3. **Architecture** - Among the architectures we have seen in the previous chapter, the ones predominantly used are client-server and peer-to-peer. In a client-server architecture, the video is routed through the server. This increases the round trip time between the source and the receiver. The variation between the round trip times can be large depending on the geographical location of the source, the receiver and the server. But the server can be used to change the video encoding to counter receiver heterogeneity. A peer-to-peer architecture has a lower RTT (round trip time) from source to receiver. But the trade off with this approach is that adding more peers to the conference causes a bottleneck in the upload bandwidth and affects scalability. Thus different topologies can cause different values of communication delay. In the case of a server, the server can repackage video (or transcode it alternatively) and add redundancy to video if the connection with the other users is lossy. The server can also drop packets to change data rates.

4. **Real-Time Protocols** - The real-time transport protocol used for video/audio delivery affects video quality. Parameters like size of packets, ordering of packets and congestion control are implemented at the transport protocol level and different protocols may give different results. More number of packets per frame make the library more susceptible to congestion as one lost packet results in a lost frame.

5. **Heterogeneity of Receivers** - In a multi-user video conferencing set up, different users have different network conditions and different hardware capability. Different hardware devices have different efficiencies in encoding and decoding video. This may be visible in the difference in the source rates and video parameters and efficiency of error handling of video from different devices.

6. **Round Trip Time** - As we had pointed out earlier, delay in audio and video causes great degradation in QoE. The delay comprises of several components such as delay in encoding at the source, delay in decoding at the receiver, delay in the

network in addition to delay at the server in the case of a client-server architecture. The delay in network from a source to a receiver (or alternatively client-server) can be evaluated using the Round Trip Time (RTT). The RTT is a characteristic of the network and is highly variable unlike the encoding decoding times which depend on the video and the hardware. The variation in the RTT is called jitter.

7. **Video Parameters** - Video quality is basically dependent on three parameters, namely frame rate (fps), video resolution and quantization parameter of the codec. The only way to combat congestion is to reduce the amount of data being sent on the network. The data rate can be influenced by changing these parameters. Packet level analysis must be done to find out which library is most robust and can handle packet losses in a better way. Also this analysis is instrumental in pointing out how smoothly the library adapts to sudden change in bandwidth at the receiver or the sender.

8. **Error Handling** - Every video conferencing library has error correction and/or retransmission policy. Retransmission is possible if the time to decode the packet is greater than the time taken by the packet to reach the receiver from the source. This imposes boundaries on the maximum delay in the network between sender and receiver. Error correction mechanisms like FEC (Forward Error Correction) [54] do not require retransmission but increase the data rate. This increase in data rate must be compensated by a subsequent decrease in video quality (in case the bandwidth is limited). Thus the error handling strategy affects the QoE.

The table 4.3 shows the correlation of the above discussed metrics with QoE parameters:

Table 4.3: Correlation of Metrics and QoE Parameters

Metrics	Parameters
Signalling Overhead	Video Quality (weakly affects video quality)
Traffic	Video Quality, Audio, Delay, Power, CPU usage
Architecture	Delay, Synchronisation, Video Quality
Real-Time Protocols	Synchronisation, Delay
Heterogeneity of Receivers	Delay, Synchronisation, Video Quality
Round Trip Time	Delay, Synchronisation
Video Parameters	Video Quality, Synchronisation, Delay, CPU usage, Power
Error Handling	Video Quality, CPU usage, Power

A framework was developed to test the robustness and analyse the performance of the library using some of the above described metrics which will be described in Chapter 5.

5 Framework

5.1 Introduction

This chapter focuses on the framework used for profiling and testing the video libraries. It discusses the design choices, the rationale behind making those design choices and the trade-offs made while designing this framework. The video conferencing library was installed on four different Android devices:

- **Google Nexus 10** - The tablet device which is the main device for this product. This device is user 1.
- **Google Nexus 7** - Another tablet device but with less powerful hardware. This device is user 2.
- **Samsung Galaxy XCover 2** - Two phones joined the conference as users 3 and 4.

The results of performance at User 1 (Google Nexus 10) are more important to us and hence the choice of device for users **2**, **3** and **4** can be any device provided it runs on Android and is connected to the internet through a Wi-Fi. The framework attempts to measure how robust the library is and tries to determine the QoE qualitatively using quantitative measurements of video parameters like source rate, frame sizes and frame resolutions [53]. The network conditions have to be carefully controlled, modified and varied to create various scenarios. To study the library under various network conditions, a controlled test bench (which is the framework) was set up which is discussed in the next section.

5.2 Controlling Network Conditions

Video conferencing libraries are advertised as showing exceedingly good performance in conditions of high available bandwidth and powerful hardware to encode/decode video with resources to spare. However adaptability of a library to different network conditions is an important performance criterion. During deployment, not all users may have access to excellent network conditions and hence a qualitative and quantitative performance analysis of the library is needed to evaluate the performance under poor network conditions. Thus, a controlled environment was set up where the effect of the conditions on the library can be observed easily with the help of packet sniffing tools like Wireshark [63]. In the controlled environment, network parameters like delay, packet loss ratio and available bandwidth are variable so that behaviour of the library under different conditions can be observed. This section describes the test bench set up for conducting the experiments. To recreate network conditions, there are predominantly three options:

1. **Traffic Shaping Router** - This is a dedicated router which shapes both incoming and outgoing traffic be varying various network parameters. However purchasing the router is costly and a new router would have to be purchased every time the hardware becomes out dated.

2. **Network Simulator** - Network simulators like NS-2, NS-3, GNS3 and OmNet++ [85] provide a complete simulation of complex networks and the topology. However interest lies in the effect of varying network conditions on the QoE of one particular application running on one particular node rather than the effectiveness of a particular topology. Due to this reason, this option is not useful for the current experiment.

3. **Network Emulator** - Network emulators allow simulation of real network conditions on a real network link between two nodes. Using network emulators, the behaviour of various applications which use the internet and the reaction of the applications when the quality of the link changes can be observed. Thus network emulators help to determine the correlation between network conditions and application behaviour.

As the goal of this effort is observing changes in application behaviour as a function of network conditions, it was decided that an emulator needed to be set up instead of a simulator. A traffic shaping router would not allow the running of packet sniffers and observation of conversations at the packet level. Also, the framework which was set up needed to be easily modifiable and upgradable in case changes were needed to it. Hence a Linux PC with an internet connection was set up as a wireless access point. Linux was chosen as the choice of operating system because some tools required for network emulation were found available on Linux directly. The PC had a wired connection which connected it to our modem connecting it to the internet. The wireless adapter of the PC was configured to provide an Ad-Hoc network to which the devices under test would be connected. Thus the PC now acted as a router for the devices under test. Both the links (wireless as well as wired) could be monitored using this set up. The test bed can be seen in figure 5.1. Mainly four Devices Under Test (abbreviated to DUT) can be seen as taking part in the experiment. **DUT 1** and **DUT 2** are the two tablet devices and they are connected to the router as the performance of the application on tablet devices is of interest. As the application is not targeted for launch on mobile phones, performance analysis of the library on the other two devices, **DUT 3** and **DUT 4** is not interesting.

A packet analyser had to be installed on the Ad-Hoc PC and analysers like *ngrep* [56] and *tcpdump* [42] were considered. *Wireshark* was chosen as the packet sniffer of choice because of the flexibility it offers although *tcpdump*, which is a command line based packet analyser based on *pcap* (packet capture library) [43] can also be used to profile the library. Wireshark supports various protocols (RTP being of main interest) and thus traces generated as UDP or TCP conversations between two different IP addresses and port numbers can be decoded using higher level protocols. Thus a conversation can be observed between higher levels protocols directly (for eg. RTP) making analysis easier. Wireshark has a huge developer and users community which can be turned into for support. It is also compatible with the *pcap* [43] in Linux and thus added functionality

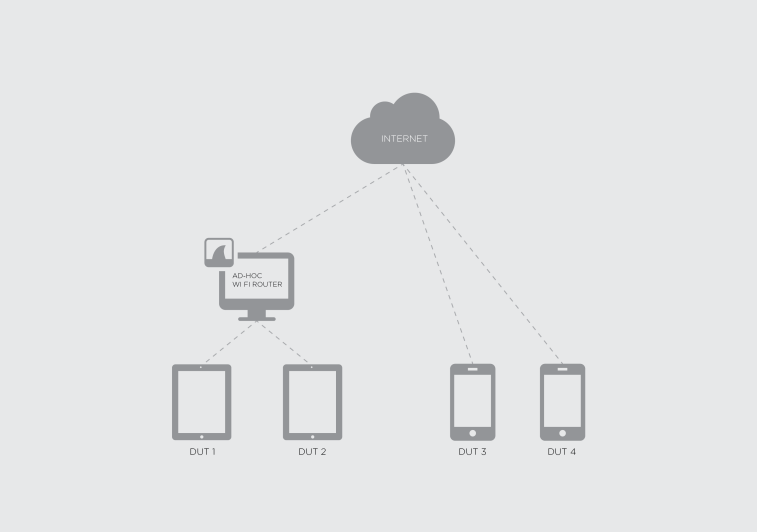


Figure 5.1: The test bench with devices connected to it

can be integrated with ease. Apart from that, there are various tools available with Wireshark such as:

- **dumpcap** - *dumpcap* is a tool which dumps the packets sniffed on the line and writes them to a file for offline analysis. The file is in the *pcap-ng* (Packet Capture New Generation) format and thus all tools in the Wireshark suite are compatible with it. Various capture and display filters can be employed with it in order to generate a trace of relevant packets. Capture filters capture packets which are useful to the analysis and reject the other packets as per the condition mentioned in the capture filter (eg. ip.addr == 192.168.1.XXX meaning only packets which are to or from the given IP address will be captured). Additional packets increase the size of the capture file and add to noise in case of statistical analysis performed on the trace. A display filter will only display the packets as per the requirements from an already captured trace of packets. Using a combination of capture and display filters, we can generate traces which are highly relevant to the data we want to analyse thus minimising noise interference [50].

- **capinfos** - *capinfos* is a tool which performs statistical analysis on captured packets. The analysis consists of various parameters like data rate, size of the trace etc. Data rates generated by any call are of specific interest for analysis [50].

- **tshark** - *tshark* is a command line based version of Wireshark. The tool is used to translate the captured file from the *pcap-ng* format to the text format thus allowing us to use text based search to analyse packets in a session [50].

Wireshark was not installed directly on the DUTs as the conference would have to be stopped every time data had to be collected. Apart from that, installing Wireshark on the router gave an opportunity to observe the outgoing trace if need be. Wireshark

provides microsecond level precision in identifying the system time at which the packet was sent or received. Although microsecond level precision is not required because as per International Telecommunication Union (ITU) recommendations, the quality of experience is adversely affected if the delay/synchronisation offset is within the order of hundreds of milliseconds.

5.2.1 Network Emulation

Linux kernel has an embedded program called *netem* [39] which provides network emulation functionality. It can emulate network parameters like delay, variable delay, loss, corruption, duplication, reordering. *netem* uses a command line tool called *tc* [40] which stands for 'traffic control'. *tc*, in combination with *netem*, can also be used effectively for bandwidth shaping. These tools provide an easy to use command line interface and can be used to shape traffic at both the wired and wireless links. This tool was used for network emulation in this test bench. As it can simulate a variety of network parameters, future changes to the test bench can be made easily.

5.3 Framework Structure and Capabilities

This section describes the design of the framework, lists its capabilities and explains how helpful and accurate it is in measuring various metrics of our interest.

5.3.1 Audio Video Delay

To measure delay accurately, the source and the receivers have to be synchronised. Except in the case of very poor network connection in which case the delay is in the order of seconds, the delay is usually in the order of milliseconds. Such a small delay, though perceptible cannot be accurately measured without any electronic equipment. An approached is discussed in literature [90] where a microsecond timer is played on the screen of a PC and the video camera of the source is pointed to it. The receiver is positioned right next to the PC screen so that both the original live timer and the video appearing at the receiver can be seen at the same time. Photos of the PC and the receiver (using a camera of reasonable quality so that the timer can be viewed clearly) are taken at regular intervals (while changing network parameters). These techniques help determine the end to end communication delay for the conference.

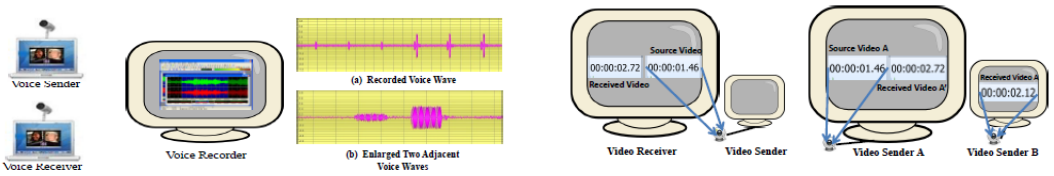
The end to end delay between two users comprises of the following components:

1. Encoding and packetization of the frame.

2. Network delay encountered by the packets.

3. Processing delay at the server (which would not be present in the case of a peer-to-peer architecture).

4. Decoding and rendering of the frame.



(a) One-way Voice Delay (b) Voice Waves Detail (a) One-way Video Delay (b) Round-trip Video Delay

Figure 5.2: Measurement set up for end to end audio and video delay [90]

This can be visualised by the following equation where 'D' stands for delay:

$$D_{Tot} = D_{enc} + D_{packetisation} + D_{network} + D_{server} + D_{dec} + D_{render} \tag{5.1}$$

An important requirement for this framework is that measurement of critical parameters should be done with a high degree of automation. The method described above is time consuming and requires a lot of manual intervention. Hence a technique which could be automated easily is preferred over a possibly more accurate but manual one. Moreover, the average and the worst case delay are two different parameters which need to be calculated and the approach of having a timer on screen is not very accurate when it comes to measuring both. The accuracy of the set up is also heavily dependent on the frequency of sampling (how frequently snapshots are taken by a camera) and hence enough useful information cannot be determined. Encoding and packetisation delay can be correlated to packet sizes and the number of packets each frame is encoded in. As two devices are connected to the router, a packet which is uploaded by one device can be seen downloaded by the other device due to the same value of the "time stamp" field or the same sequence number in the RTP header. The network delay can be obtained from subtracting the detection times of both the packets in Wireshark.

Audio data is far less bandwidth insensitive as compared to video data and does not face problems of frame level delays. Frame level delays are different than packet level delays as a frame may be delayed even if one of the packet constituting the frame is delayed. However, in cases where the user wishes to keep his video private and chooses to communicate only using audio, the audio delay becomes an important parameter and needs to be measured. Measurement of audio delay encounters the same problem as measurement of video delay as human ears are not sensitive to delays in the level of microseconds. Unlike measurement of video delay where a still picture of one of the video frames can be used to measure the delay, a still audio frame is more difficult to capture. A method has been discussed in literature [90] to determine audio delay accurately which uses a digital audio analyser [6]. A ticking sound is played at the source and the audio is heard at the receiver. The source and receiver are placed nearby and the original audio as well as the audio from the receiver are captured by the audio analyser. The tick at the source marks a distinct spike in the audio wave and the tick from the receiver marks a more subdued spike. By comparing the time different between the spikes, the audio delay between the source and the receiver can be almost accurately measured.

This approach has the same drawbacks as discussed in the case of measuring video delay. While running the experiments, even a small noise like a tap on the table can create enough noise to make distinction between the original tick and the table tap impossible. Hence the framework uses a similar approach as discussed for measuring video delays. Audio packets can be differentiated from video packets due to the presence of a mark bit in them. Some video packets have the mark bit set which is used to determine frame boundaries. Using this, audio payload can be differentiated from video payload and audio packets can be identified. Following isolation of audio packets, unique packets can be detected due to their unique time stamp or sequence number and the delay determined by subtracting the detection time of both the packets at different devices.

5.3.2 Audio Video Synchronisation

Audio Video synchronisation plays an important part in the QoE as discussed previously. By superimposing the results of video delay and audio delay measured at the same time, empirical evidence can be collected which can determine whether large scale audio video synchronisation is absent or not. Even if one of the media is delayed more than the other, the library can still delay the other one in order to ensure audio video synchronisation at the expense of total delay. Audio video synchronisation may be affected by other competing data, or more of CPU resources being dedicated towards competing applications. This approach however is useful in determining whether architectural fallacies in the infrastructure are responsible for the lack in synchronisation or not. This approach has less visibility as compared to the direct approach where a video will be playing a clock and making a timer tick simultaneously and the delay will be measured using the approaches discussed in the previous subsection. However as discussed above, automation is not possible for this approach.

5.3.3 Identification of the Server and subsequent analysis

In case of a client-server library (S/C), the geographical location of the server plays an important part in determining the latency of the video signal. If the server is located in a place far away from the users, it adds a considerable latency to the video signal as compared to the case when the server is geographically near. The IP address of the server is found out by identifying RTP streams originating from one of the devices. The destination address of the packets in the media stream is the address of the server. The topology of the conference can also be found out using this approach. If all the devices send media streams to one IP address (or different IP addresses within the same subnet), then that IP address is the address of the server and the topology of the library is S/C. If each device sends its media stream directly to the other device then the topology can be identified as peer-to-peer. Using IP geo-location tools on the internet [7] the location of the server is found out. The IP address of the server is also used to filter out relevant conversations using Wireshark (as media traffic from each user will be sent directly to the server).

If the video is relayed using a server, the server may play an important part in maintaining the QoE and congestion control. The role of the server if any, in shaping the media stream can be determined by manual observation of the Wireshark trace. Packets

uploaded by one device can be tracked while being downloaded at the other device using techniques discussed earlier. If the packets are reshaped, the value in the 'timestamp' field in their RTP header would be different. The size of the uploaded and downloaded packets may be different, suggesting reshaping at the server. In case of transcoding, the payload type of the uploaded packets will be different from the downloaded ones. The packets which are missing (are seen in the upload stream but not in the download stream) have been lost either due to random packet loss in the internet or they have been dropped by the server to prevent overload of packets at the receiver. As frames can be detected in the video stream (using 'Mark' bits - something which will be discussed in the later part of this section), information can be obtained whether an entire frame has been dropped or reshaped. This data can be used to determine the role of the server in congestion control.

5.3.4 Packet Size, Frame Size and Frame Rate

If data is encapsulated in larger packets, it is more susceptible to packet loss as lost of one packet would lead to loss of a large amount of data. Each RTP packet is identified by decoding the media stream as RTP in Wireshark and the sizes of the packet are known in bytes. The average size of packets in a trace can be found out during offline analysis.

Each RTP packet from a source has a unique sequence number. The sequence number of each packet is used for ordering and helps detect out of order packets. The "timestamp" field carries the time stamp of the packets and is used to determine the exact time at which each packet has to be rendered. The timestamp values are decided at the time of session creation and are exchanged at the beginning through the session management protocol which is in use. Once a session has been created, senders and receivers exchange synchronisation information using Real-Time Transport Control Protocol (RTCP) senders and receivers report (SR and RR).

Each user who is a part of the call has a unique Synchronisation Source Identifier (SSRC) which is decided at the time of session creation. The origin of each packet in the call can be determined using its SSRC field. Using a combination of SSRC, the sequence number and the timestamp value, every packet can be uniquely identified and tracked as it is uploaded by a device and downloaded by the other.

An RTP packet has a 'Mark' bit in the RTP header. In case of an audio stream, the 'Mark' bit denotes the beginning of a talk spurt. Usually audio streams do not enable the mark bit. In case of a video stream, the 'Mark' bit denotes the end of an frame. Figure 5.3 shows the RTP header where all the fields like Mark, SSRC, sequence number, payload type number, time stamp etc. can be seen. Packets within the same frame have the same time stamp value. Only one of the packets with the same time stamp has the Mark bit enabled and this packet signifies the end of a frame. In case of a single packet frame, each packet will have the Mark bit enabled.

By observing the number of frames detected in an n second trace, the frame rate of the video can be determined. If a frame is comprised of more than one packet, loss of a single packet may cause loss of an entire frame resulting in choppy video quality. Large scale frame losses also affect audio video synchronisation.

The PT field is the payload type. The Internet Engineering Task Force has a fixed

payload number for standardised media types. However the registration format also allows for dynamic registration where newer payloads like VP8 encoding for video which is not yet standardised can be incorporated. By identifying the payload type number for the video stream, the use of specific codecs, if any, can be determined (only if the call uses standardised media types). However even if the call does not use standardised media types, different payload type numbers in use for audio and video streams can be determined which stay constant throughout the session. These payload type numbers help discriminate an audio stream from a video stream easily.

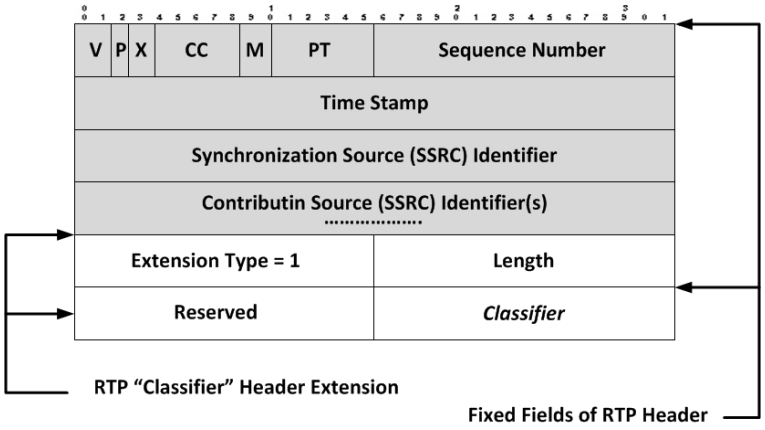


Figure 5.3: Header for Real-Time Protocol

5.3.5 Video Quality

An objective measure of video quality is essential for gauging the video quality in a standardised manner. Objective methods like bVQM [2] (Batch Video Quality Metric) for establishing video quality have been discussed previously. The algorithm compares two videos, one which is injected into the input of the sender and the received video at the receiver and comes up with a score which is a reflection of its fidelity as compared to the original one. A score of 0 or near to it suggests very good quality and a score nearing 1 suggests bad quality. In exceptional cases a score of more than 1 may be seen suggesting worse (completely unusable) video quality.

For successful usage of the algorithm, the same video source should be used for every run of the experiment for standardisation purpose. A virtual software camera allows streaming of a pre-recorded video as a captured video. A screen recorder would have to be used for isolating the received video. However in a multi-user conference, videos from all users are rendered simultaneously on the screen and hence isolating one particular video is difficult. Also a suitable virtual camera was not found for Android.

However linear correlation has been observed between bVQM and MOS (Mean Opinion Score) [53]. Mean Opinion Score is a score derived from showing users the video and asking them of its quality. A score of 1 suggests bad video quality, 2 = poor, 3 = fair, 4 = good and a score of 5 suggests exceptionally good video quality. The correlation has been observed to be valid in the range of MOS between 1 and 4. Hence the mean opinion

score based approach was chosen for comparing video quality. However as standardisation of video sources was not possible, video quality was judged by observation to be either Good, Bad or Tolerable. A good video means that motion and lip movement is seen clearly. A tolerable video means that motion is present to a low degree and a bad video means that very little or no motion is seen.

Video quality can be empirically determined using frame sizes and frame rates. If the average frame rate of a call (sampled over a known period) turns out to be high, the video will have high motion content. If the frame sizes are high, the video carries high resolution. A video having a high resolution frame but a frame rate of 12-15 FPS may seem good to a user as also a frame having relatively low resolution but a higher frame rate. By plotting both frame sizes and frame rates in time, it can be conjectured with whether the video quality is 'Good' or 'Bad'.

5.3.6 Bandwidth Usage of a Call

Measuring the total traffic for each user is an important metric as it reveals how efficiently a video conferencing library uses the bandwidth available to it. The bandwidth required for any call increases as more number of users connect to a call. However as the download bandwidth is limited, the library may try to improve scalability by reducing the bandwidth usage of the call by employing traffic shaping techniques. Measuring the bandwidth usage of a call reveals whether any traffic shaping techniques were used. It can be used to determine how effective the traffic shaping strategy is and whether more users can be added to the call with the given bandwidth or not. In order to differentiate between shaping of video, audio and signalling traffic, the conference is profiled under three different scenarios as follows:

1. **Full conference** - In this mode the traffic consists of video, audio and signalling traffic. This mode measures the audio, video as well as signalling bandwidth usage.

2. **Audio** - In this mode, the video is disabled and the conference is run as a audio conference with signalling overhead. This mode measures the audio and signalling bandwidth usage.

3. **Signalling** - In this mode neither audio nor video is enabled and all the users only exchanging signalling messages with the server. This mode measures the only the signalling bandwidth usage.

The video bandwidth usage can be determined by subtracting the audio mode (audio + signalling) bandwidth usage from the full conference (audio + video + signalling) bandwidth usage. Similarly audio, and signalling bandwidth usage can be determined. The following equation shows the dissection of the traffic into video traffic, audio traffic and signalling traffic.

$$T_{Tot} = T_{Video} + T_{Audio} + T_{Sig}$$

Bandwidth usage, along with the frame rates and frame sizes can also be used to estimate video quality. A call with higher frame sizes and frame rates is likelier to have

high bandwidth usage as compared to a call with lower frame sizes, frame rates or both. When measured for scenarios (network conditions, configurations or different libraries), these three parameters can be used to estimate whether the higher bandwidth usage is because of higher frame sizes or frame rates or both.

5.4 Summary

This framework is mainly designed to promulgate our understanding of how video conferencing technologies work and to determine how much resources are used by them. This framework enables us to identify bottlenecks in realising a multi-user video conference on an embedded computing platform and to use our developed knowledge to make a decision on which group of choices suit our requirements the best. It mainly profiles the performance of the video conferencing library at run-time by monitoring data such as audio/video delay, frame/packet sizes, frame rates and bandwidth usage of the call under different network conditions. The framework was written using a combination of shell and python scripts and by using **netem** and **tc** [40] for traffic shaping.

Different experiments can be set up using this framework to test different aspects of the library. The next chapter discusses how the experiments were designed and analyses the results. The interpretation of the results is also discussed at length allowing us to draw conclusions about performance of the libraries. The experiments were performed in the months of July, August and September 2013 in Delft, The Netherlands. All the devices were tested in the Netherlands itself as the initial launch of the product is the Dutch market.

6 Experiments and Analysis

This chapter describes the experiments which were designed using the framework and discusses the results generated after the framework was applied to two different libraries, Library 1 and Google Hangouts. These experiments are designed to test and understand various functionalities of a video conferencing library. Apart from the experiments, various components used in the design of the framework provide an interesting insight into the behaviour of a library, like its topology and pre-connection process. This chapter also discusses these findings derived in the course of applying the framework.

6.1 Pre-experimentation modification

Library 1 is different from Google Hangouts in the way that Library 1 is available to us as an Android software development kit (SDK) along with an application programming interface (API). The API could be played with to tweak certain functionality. Library 1 had multi conferencing capabilities but rendered only one user at a time. Hence in order to check whether it had true multi-conferencing capabilities, the functionality was changed to allow it to render up to three more users at a time (one plus three other participants in the conference making the total number of participants four).

The size of the rendering window was fixed at 320 x 240 pixels so that up to 4 users could be rendered simultaneously. The changes made were successful as the library could be compiled for Android and a true multi conference video call was possible as all the 4 users could be rendered on the screen. This provided an accurate estimate of the CPU usage in the case of a multi-user call as video rendering famously uses a lot of CPU.

Library 1 could be configured by application developers for different values of frame rates, frame resolutions and bit rates. For example if the library is configured for a maximum of 25 FPS (frames per second), the frame rate of the video should ideally never exceed 25, thus acting as an upper cap. Google Hangouts is however a software application and hence it is not possible for a user to configure video parameters.

Any questions during the development/modification were asked to the vendors and technical support was available at a short notice. Availability of technical support was a plus point as the development of newer products based on the library can be accelerated in the presence of good technical support.

Library 1 was found to contain an Acoustic Echo Cancellation module (AEC) and a Noise Suppression module (NS) and hence it was decided that it meets certain mandatory requirements for realising a video conference with a decent quality of experience (QoE).

The following sections discuss in detail how each experiment was designed using the framework and how the generated results were interpreted to arrive at a useful conclusion.

6.2 Configuration Experiment

This experiment mainly tries to understand how the library behaves differently under different configurations. As discussed earlier, Library 1 can be configured for different values of video parameters like bit rate, capture resolution and frame rate. This experiment also tries to understand how the library adapts to change in traffic load resulting from addition of more users. Hence this experiment was run under 9 different configurations for 3 different group sizes of 2 users, 3 users and 4 users. The capture resolution for all configurations was kept constant at 320 x 240 pixels. For each configuration and group size, this experiment measures the following parameters:

1. Audio, video and signalling bandwidth usage at DUT 1.
2. Frame sizes in bytes and in number of packets per frame for both DUT 1 and 2.
3. Packet sizes for both audio and video for both DUT 1 and 2.
4. System level packet delay for both audio and video packets from DUT 1 to 2 and DUT 2 to 1.
5. CPU usage for DUT 1

The configurations of the library for this experiment are summarised in Table 6.1.

Table 6.1: Configurations of Library 1 for the configuration experiment

Configuration	Call Sizes		
5 FPS, 256 Kbps	2 Users	3 Users	4 Users
5 FPS, 512 Kbps	2 Users	3 Users	4 Users
5 FPS, 1024 Kbps	2 Users	3 Users	4 Users
10 FPS, 256 Kbps	2 Users	3 Users	4 Users
10 FPS, 512 Kbps	2 Users	3 Users	4 Users
10 FPS, 1024 Kbps	2 Users	3 Users	4 Users
15 FPS, 256 Kbps	2 Users	3 Users	4 Users
15 FPS, 512 Kbps	2 Users	3 Users	4 Users
15 FPS, 1024 Kbps	2 Users	3 Users	4 Users

6.2.1 Details of each configuration

Each configuration consists of 3 parameters. This subsection discusses how each configuration was selected for this experiment.

1. Frame Rate - Three different frame rate values were chosen for experiment 1. A frame rate of 5 FPS was chosen to observe the behaviour of the library in low frame rate conditions. The frame rate was then increased to 10 FPS and 15 FPS in order to observe how the behaviour of the library changes with frame rate. Although

a frame rate of 15 FPS is low as compared to High Definition video (60 FPS) or even standard definition video (30 FPS, 24 FPS) it was chosen as a representative value. A typical house hold video conference has less motion (compared to out door surroundings or sports feed) and a frame rate of 15 FPS was observed to provide 'Good' video on observation. In subsequent experiments, more data points with higher frames per second will be tested.

2. Bit rates - Three representative values were chosen for conditions of low, medium and high bit rate. For low bit rate conditions, the bit rate was fixed at 256 Kbps. For medium conditions, the bit rate was fixed at 512 Kbps and for high bit rate conditions, the bit rate was fixed at 1024 Kbps.

3. Number of users - As we had 4 Android devices on which we could run our tests, we had 3 different conference configurations: - two users (which constitutes a two way conference), three users (a basic multi-user conference) and a four user conference (a multi-user conference with higher population and hence higher traffic at each user).

User 1 uses the Google Nexus 10 and always launches the conference. User 1 stays connected to the framework for this and for all other experiments. User 2 always uses a Nexus 7 and is always the second user (which means that in a 2 user call, User 2 always uses a Nexus 7). Users 3 and 4 use Samsung Galaxy XCover 2 phones and are not connected to the framework.

As video traffic is subject to changes over time with motion estimation getting efficient with time, the full conference is run for 10 minutes and the average traffic load measured. The audio only conference is run for one minute and the signalling only conference is run for 30 seconds.

6.2.2 Results

This part of the section discusses the results after the framework was applied on Library 1.

For each experiment, the framework always begins with finding out the topology of the library. It was observed that all the users in the conference uploaded their media packets to one particular IP address. That particular IP address remained constant throughout the duration of the call. This meant that the IP address was the address of the server for that particular call session. Occasionally, different servers were used for different sessions but they remained the same throughout each session. Hence it is observed that the library is a client-server (S/C) library.

RTP packets of payload type 76 were observed in the framework. This means that the library uses RTCP for providing feedback via senders and receivers reports. The payload type 76 is reserved for RTCP so that data and control packets can be distinguished. Packets of 2 other payload types were observed where both the payload types were within the dynamic range. This meant that the audio/video encoding formats used were defined before the start of the session using some conference control protocol and were not standardised with a payload type number as defined by the Internet Assigned

Numbers Authority (IANA) [25]. In a conference where the video was switched off, packets of only one payload type were observed and hence it was concluded that this payload type corresponded to audio. The other payload type was thereby concluded to be the number for the video codec. As the audio packets were not arranged into frames, it was concluded that a frame based audio codec was not used.

Bandwidth Usage of Call

The audio, video and signalling bandwidth of the call was measured for each configuration using the framework using the technique described in the previous chapter. For video bandwidth, the call was sampled for 600 seconds so that it would have enough time to stabilise and any undue variations in traffic would smooth out. In order to further mitigate the random effects of motion, the cameras of all users in the conference were made to face plain white paper. Bandwidth usage for audio and video for some of the configurations is plotted in Figure 6.1.

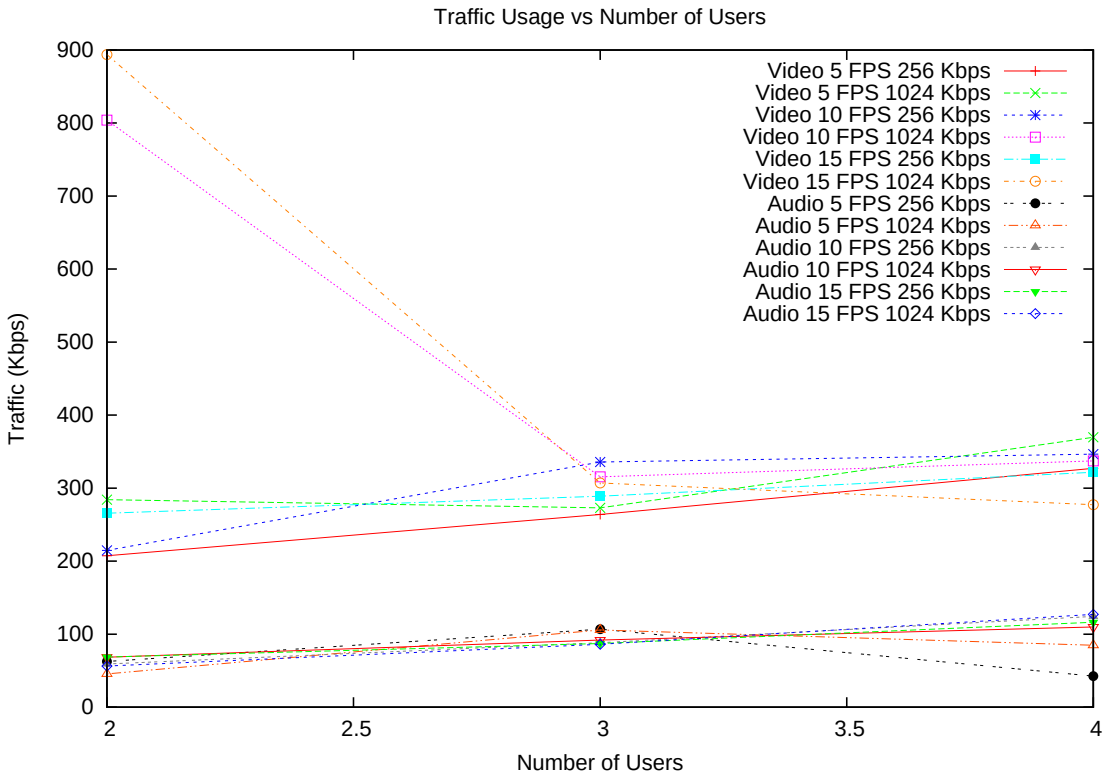


Figure 6.1: Bandwidth Usage vs. Number of Users (Device 1)

The different configurations of the library all have differently configured video streams and not audio streams. Hence no major changes in the audio traffic profile should be expected. Figure 6.1 corroborates this hypothesis as the audio bandwidth usage for different configurations is nearly similar. The bandwidth usage increases monotonically as the number of users in the conference increases. No evidence of traffic shaping in

audio traffic suggests that audio traffic has a high priority and uses the same bandwidth irrespective of video quality. No significant change in size of audio packets was observed for different configurations as well as different call sizes.

An interesting fact is that silence suppression was not visible in Library 1. Audio packets were observed to flow between users even though they were not speaking. Although this approach consumes more bandwidth, it helps the audio session between the two users remain alive and maintains UDP bindings. Thus in case of a user speaking suddenly, overhead in establishing UDP bindings is eliminated. This also helps to transmit background noise at the user so that a complete conference experience can be generated. Comfort Noise (CN) packets can be observed easily in the RTP stream [94] and in this case, they were not observed. Comfort noise packets are sent so that the users do not mistake periods of silence in the conversation for loss of transmission.

It was observed that the bandwidth usage of signalling remained nearly constant (between 2 Kbps and 3 Kbps) for all configurations and group sizes. As the library had a client server architecture, each client connected to the server and maintained connection with it irrespective of the group size. The signalling overhead was observed to consist of periodic RTCP messages (Sender's and Receiver's reports) [86] for signalling information from server to client. Apart from that, encrypted messages from client to server (over TCP) were observed. This signalling overhead is too minimal to be considered bandwidth intensive and is not likely to become a bottleneck in the case of increasing group sizes.

Video traffic was observed to show signs of traffic shaping. At low bit rates (256 Kbps) and low frame rates (5 FPS), video call was observed to consume less bandwidth as compared to configurations with higher bit rates and higher frame rates (e.g. 10 FPS 1024 Kbps and 15 FPS 1024 Kbps). For the low bandwidth configurations, the total bandwidth usage was sufficiently low and more users could be incorporated into the call without employing any traffic shaping. This can be observed due to the monotonically increasing video traffic profile of these configurations. However in the case of the configurations requiring high bandwidth, evidence of traffic shaping techniques was observed. For these configurations, the bandwidth usage of a 3 user call was observed to be lower than a 2 user call suggesting that some traffic shaping took place. This traffic shaping took place most likely by changing the configuration dynamically, i.e. by changing bit rates, frame rates, frame sizes, frame quality or a combination of these options. Also the video traffic profile for all configurations for group sizes of 3 and 4 was similar. This suggests that for a 3 user call, the library switches to a default low bandwidth configuration, irrespective of the original configuration.

To investigate the traffic shaping more carefully, the frame sizes for both the devices was observed for all configurations. Figure 6.2 plots frame sizes for both DUT1 and DUT2 for a frame rate of 5 Hz and Figure 6.3 plots the same for a frame rate of 15 Hz.

The first observation that can be made from both the figures is that the frame sizes for DUT 1 are very small as compared to DUT 2. This can be explained by the different hardware configurations of both the devices. DUT 1 is a Google Nexus 10 running on an Exynos 5 Dual chipset. This chip has an ARM Dual-core 1.7 GHz Cortex-A15 CPU, a Mali-T604 GPU along with a RAM of 2 GB. Opposed to that DUT 2 which is an Asus Nexus 7 running a NVIDIA Tegra 3 processor core comprising of an ARM Quad-core 1.2 GHz Cortex-A9 CPU, a ULP GeForce GPU along with a 1 GB RAM. Studies have

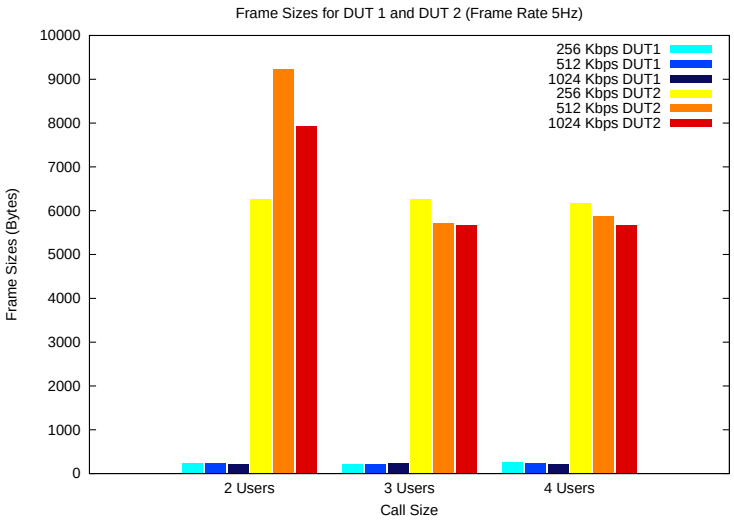


Figure 6.2: Frame Sizes vs. Number of Users - DUT 1 and 2 for 5 FPS

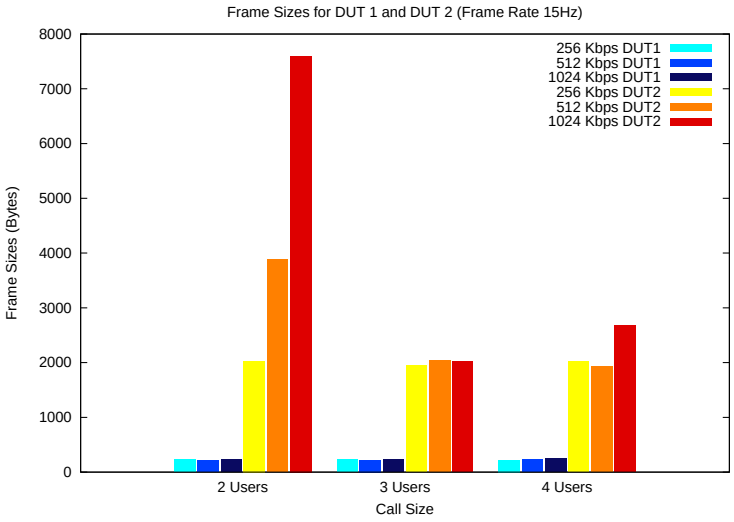


Figure 6.3: Frame Sizes vs. Number of Users - DUT 1 and 2 for 15 FPS

shown Exynos 5 to be much faster in performance as compared to Tegra 3 and this coupled with larger RAM sizes enables DUT 1 to achieve higher compression for the same quality.

Very low frame sizes for DUT 1 as compared to DUT 2 are indicative of the fact that the compression of DUT 1 is far more efficient than on DUT 2. Both DUT 1 and DUT 2 have encoding and decoding support for H.264 and only decoding support for the VP8 codec. DUT 1 is however running Android 4.3 which has VP8 support which is not present in Android 4.2.2 which is running on DUT 2. This may lead us to believe that VP8 might be the codec used for library 1. However it cannot be determined with certainty as the technical specifications of library 1 are not disclosed.

Bandwidth usage for DUT 1 consists of bandwidth used by video uploaded by DUT 1 to the server as well as video downloaded by DUT 1 from the server (which is uploaded by DUT 2 and other users in the conference). It can also be observed in Figure 6.2 that the frame sizes of DUT1 remained relatively constant for call sizes of 2, 3 and 4 users, for frame rates of both 5 Hz and 15 Hz. This suggests that the bandwidth shaping techniques employed by Library 1 affect the video stream of DUT 2 more than that of DUT 1. DUT 2 generates on an average higher bit rates as compared to DUT 1 and hence video traffic shaping at DUT 2 is likelier to bring more results.

Frame Sizes

It can be seen in Figure 6.2 that the frame sizes for DUT 2 reduced for a 3 user call as compared to a 2 user call. As the maximum resolution is fixed, the frame sizes depend largely on compression. It can be observed that for a configuration of 256 Kbps bit rate, the frame sizes for DUT 2 are relatively constant for all call sizes suggesting that the bandwidth usage of that configuration is too low to warrant any traffic shaping. However for higher bit rates, the frame sizes readjust to more or less the same for a configuration with bit rate of 256 Kbps. This suggests that the reconfigured video has a maximum bit rate of 256 Kbps. This finding is corroborated by looking at Figure 6.3 as it can be observed that for DUT 2, frame sizes remain relatively constant for a bit rate of 256 Kbps but for higher bit rates, they readjust to become more or less equal to the frame sizes generated for a bit rate of 256 Kbps. This strongly suggests that in order to avoid congestion, library 1 reconfigures the maximum bit rate of a video to 256 Kbps. When the results generated for a frame rate of 10 Hz are considered, they are observed to be consistent with this hypothesis.

It was also observed that the frame sizes generated by library 1 for DUT 1 remained more or less constant for allowable bit rates of 256 Kbps, 512 Kbps and 1024 Kbps for varying frame rates. For the same video resolution, the configurations allowing higher bit rates generated similar frame sizes for configurations allowing a maximum bit rate of 256 Kbps. This suggests that DUT 1 is able to achieve the same amount of compression for all configurations irrespective of the available cap. Sensing this, library 1 encodes videos for a lower bit rate thus ensuring robustness against congestion and minimising bandwidth usage.

Thus it has been determined that video source rate shaping takes place when call sizes are greater than 2. Hence in order to determine the effect of different configurations (especially frame rates and bit rates) on frame sizes on different devices, only the results of a 2 user call are considered. Figure 6.4 plots frame sizes for DUT 1 and 2 for different configurations for a call size of 2.

The graph validates our finding that DUT 1 achieves nearly the same amount of compression for all configurations. DUT 2 has higher frame sizes and it has been observed that the frame sizes increase as the upper limit on bit rates increases. The video resolution for all configurations is the same meaning that the increase in size is due to increase in quality. For frame rates of 10 Hz and 15 Hz, the frame sizes increase monotonically for increase in maximum bit rate. hence DUT 2 delivered higher quality video when the bit rate limit was increased. For a frame rate of 5 Hz, the frame size decreases

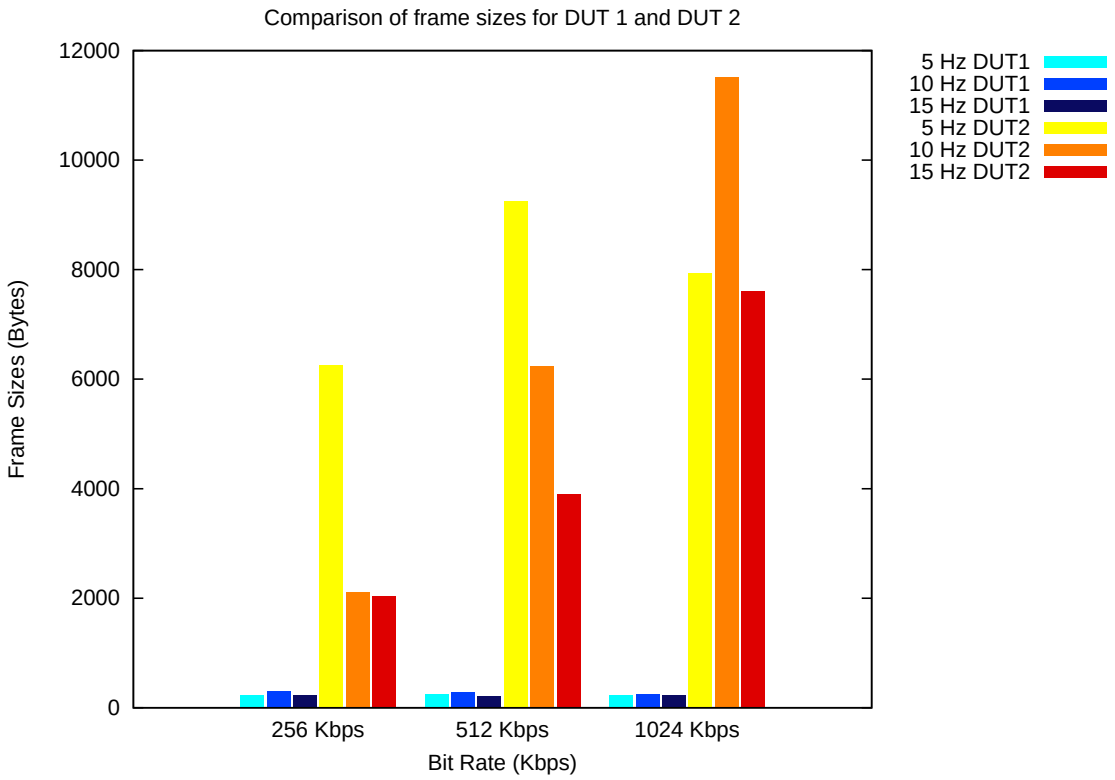


Figure 6.4: Frame Sizes for different configurations for a 2 user call

when the maximum achievable bit rate is 1024 Kbps. This suggests that some resolution is dropped in order to accommodate higher quality while keeping the data rate on the network within allowable limits.

Another noticeable observation is that frame sizes for a frame rate of 5 Hz are generally higher than the frame sizes for higher frame rates. When the frame rate is lower, the motion content in the video is low and hence more number of I frames are present as compared to P frames. As I frames require more data compared to P frames, the average frame size in the call increases.

Delay

The system level packet delay was measured between DUT 1 and DUT 2 and vice versa for both audio and video packets. It was observed that delay for audio packets was the same as that of the video packets, for both DUT 1 to 2 and vice versa. Hence no large level delays between audio and video are observed for DUT 1.

It was also observed that the delay for video (and also audio) packets from DUT 1 to DUT 2 remained around between 145 to 155 ms for all configurations and all call sizes. However, the delay for video packets from DUT 2 to 1 was observed to be around 158 to 173ms. DUT 2 generates very high traffic as compared to DUT 1 (due to much higher frame sizes) and this high traffic increases congestion on the link leading to slightly higher

delays for both video and audio packets. Although this delay is less than the 300 ms which is prescribed by International Telecommunications Union as the necessary delay for interactive communication, it can still be further reduced by reducing the frame sizes (by efficient use of hardware). This delay in both the paths (DUT 1 to 2 and DUT 2 to 1) does not adversely affect the QoE and is too small to be noticed by the user.

The higher values of delay from DUT 2 to DUT 1 suggests that video from DUT 2 is more susceptible to jitter due to the higher bandwidth usage. This affects the end to end delay (which is experienced by the user in the following way) as it may delay the frame by a value corresponding to the maximum delay faced by a packet. Each frame may be composed of more than one packet. This can be seen in Figure 6.5 which plots the number of packets for each frame.

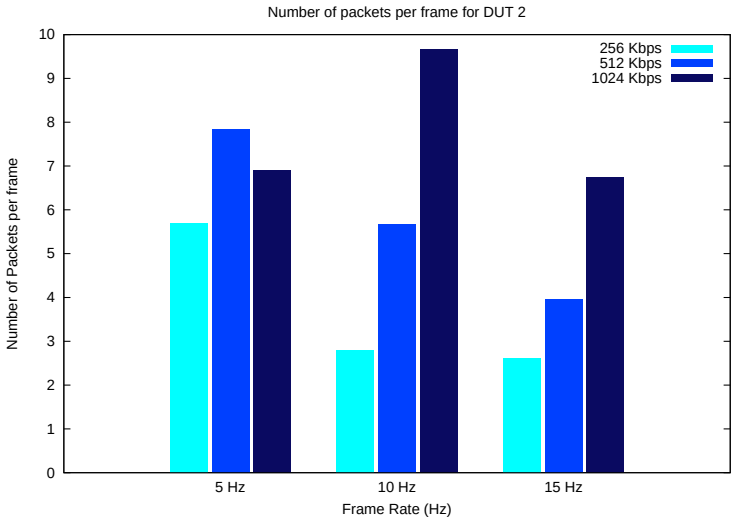


Figure 6.5: Number of packets per frame for a frame from DUT 2

The delay values measured by the framework are the average delays for each call session. The delay faced by each individual packet may be larger than the average delay. If a packet comprising a frame is delayed, it means that the entire frame is delayed. If the delay for any one packet causes the frame to miss its rendering deadline, the entire frame is useless. This causes choppy video at the user and reduces the video quality at his end. Hence the higher the number of packets per frame, the more susceptible the frame is to delays. Modifications can be made to the framework to directly deduce the frame level system delay, which is the maximum delay faced by any packet comprising that frame.

CPU Usage

When only one user is connected to the server (before the other users join), the CPU usage of the library on DUT 1 is observed to be nearly constant irrespective of the configuration. Encoding is a function of CPU usage and hence no variation in CPU values suggests two things:

1. All configurations are encoded using the same target bit rate.

2. Hardware acceleration is used.

Apart from that, for constant call size and frame rate, the CPU usage of the library was nearly constant for different values of target bit rate. CPU usage does not vary much for different configurations and this is strong evidence that library 1 uses hardware acceleration for DUT 1.

The CPU usage did increase marginally when the frame rate was increased. When the frame rate increases (frame rate increases for all users, a configuration where frame rates for different users are different is not tested), the video of the other participant which is rendered on the screen has to be refreshed more number of times per second resulting in higher CPU usage.

It was observed that the CPU usage increased when the call sizes increased. For a call size of 2, only one (the other participant) is rendered on the screen whereas in the case of a 4 user call three other users are rendered on the screen. Hence call size increases the rendering area and hence CPU usage increases for increasing call sizes. Figure 6.6 plots the call sizes for various configurations for frame rates of 5 Hz and 15 Hz and bit rates of 256 and 1024 Kbps.

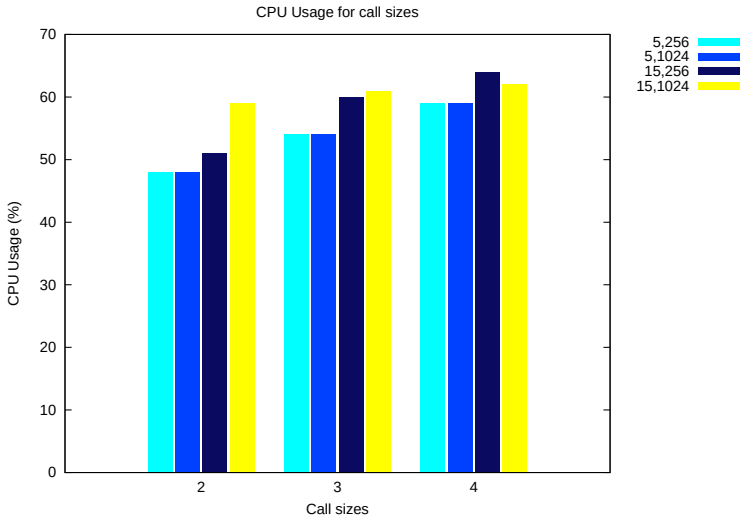


Figure 6.6: CPU Usage for Device 1 for various call sizes

6.3 Effect of Packet Loss and Packet Corruption

This experiment is designed to study the effects of packet loss, packet corruption and network latency. DUT 1 and DUT 2 engage in a two way call and are connected to our ad-hoc router and the conference is run in a video only mode.

This experiment measures the following parameters:

1. The bandwidth used by the video stream uploaded from DUT 1.

2. Frame sizes at DUT 1 and 2.

3. Number of packets per frame for DUT 1 and 2.

4. System level packet delay for video packets from DUT 1 to 2 and DUT 2 to 1.

The experiment was performed in network conditions with four different values of packet loss ratio, **0%** to represent good network conditions, **2%** to represent low loss conditions, **10%** to represent medium loss conditions and **25%** to represent high loss conditions. Measurements were taken immediately after the network conditions changed (so that robustness could be observed) and the video data was obtained for a trace of duration 120 seconds.

One instance of the experiment was run by posturing the cameras of the devices in front of a plain paper so that the variable effects of motion could be mitigated. For this instance, the packet loss ratio and packet corruption ratio were changed. Another instance was run with the camera pointing to normal surroundings allowing for random motion effects to influence traffic rate. Table 6.2 summarises the configurations of the library as well as the network conditions they were subjected to.

Table 6.2: Configurations of Library 1 and network conditions for observing the effects of packet loss

Configuration	Packet Loss/Corruption Ratio			
5 FPS, 1024 Kbps	0%	2%	10%	25%
15 FPS, 256 Kbps	0%	2%	10%	25%
15 FPS, 1024 Kbps	0%	2%	10%	25%
25 FPS, 1024 Kbps	0%	2%	10%	25%

Random packet loss can cause degradation in the quality of experience as even one lost packet may cause loss of a complete frame. There are mainly three techniques used to guarantee performance in lossy conditions:

1. **Retransmission** - The lost packets are retransmitted.

2. **Source Rate Control** - Reduce the source rate as packet loss may be caused by congesting the network with a high number of media packets.

3. **Adding Redundancy** - Temporal, spatial or quality redundancy is added to the video or error correction (or lost packet recovery) techniques like Forward Error Coding (FEC) are used.

An approach to lossy conditions can be sending duplicate packets so that even though some packets are lost, the redundant packet can make up the frame thus reducing the chances of a frame being completely lost. Such redundant packets can be seen in the trace and identified by the same time stamp.

Some systems decide that the loss of packets is due to congestion of the network due to flooding of media packets. Hence the source rate is reduced to avoid congestion and

thus reduce packet loss. Some networks employ QoS at the router (using the diffserv mechanism in IPv4 and IPv6) to ensure that time sensitive media packets receive priority and are not lost or delayed due to congestion. However the framework is incapable of detecting whether QoS at the router is being used to control video overload.

Yet another approach is adding spatial redundancy to each frame where the frame can be reconstructed at the receiver even with the loss of a few packets. As opposed to spatial redundancy, the frame rate may be increased, thus adding temporal redundancy so that good quality video may be received even in the loss of few frames.

One of the most common techniques for lost packet recovery is Forward Error Correction (FEC) where a k source packets are encoded into a group of n source packets where any k of the n packets can be used to generate all n packets. The other packets which are used for error correction are called parity packets and in order to reduce the bandwidth usage, the library may only send a subset of the parity packets and send the later in case they are essential to the receiver. This approach although it saves bandwidth may increase end to end delay from the source to the receiver. The parity packets can be added at the Application Data Unit level (which means additional RTP packets) or at the Protocol Data Unit level (increase in the number of underlying UDP packets). They may be used individually or a combination of these approaches can be used to ensure robustness of a higher degree but at a higher level of complexity. Use of these approaches mark an increase in the source rate and more information about the use of these approaches can be found out by tracking the frame rates and frame sizes.

A video predominantly consists of three types of frames:-

1. **I Frames** - These frames are encoded using blocks in the same frame itself. These are reference frames and can be decoded by themselves without the need of any other frames.

2. **P Frames** - A P frame of predicted frame is encoded using previously encoded frames as reference frames (both I and P frames may be used for reference).

3. **B Frames** - These frames are encoded using both previously coded and future reference frames. Hence a B frame may be encoded after a future reference frame, several frames into the future is encoded.

P frames require considerably less data to encode as compared to I frames, but the encoding process demands more RAM and computational complexity. Decoding each frame requires that the decoder has access to the previous encoded reference frames. In case one or more of the reference frames is lost, the P frame can be decoded with low quality or not decoded at all. Due to this dependence, the loss of a single packet may cause loss in quality or complete loss of more than one frame in the video.

B frames generate much less data per frame as compared to both I and P frames but also increase the latency in the video upto limits which are not tolerable in the case of interactive applications. Hence video conferencing applications seldom use B frames and use video composed of purely I and P frames. Loss of P frames or B frames causes less damage to video quality as compared to the loss of I frames.

In case of loss of decoding context due to loss of some frames (due to packet loss), the decoder (receiver) demands more I frames from the encoder (source) through the use

of RTCP FIR (Real-Time Control Protocol Full Intra Request) message as an I frame is needed to repair the decoding context of a prediction chain. Hence lossy conditions may cause the receiver to demand more and more I frames. I frames generate considerably more data than P frames and the average size of frames in a trace might increase if the number of I frames increase. As discussed, this behaviour may be detected conclusively by decoding the RTCP RR (Receiver's Report) for presence of FIR messages. The framework at this point is not capable of decoding RTCP messages but with some modifications, this functionality can be added.

In the end, it can be summarised that the ultimate goal of these techniques is to reduce the likelihood of packet loss and to allow media flows to use the available bandwidth efficiently.

In case of Library 1, no information is available on what error correction techniques are used and how robust they are in face of packet loss. This experiment applies our framework to the library in order to see how does error redundancy manifests itself at the frame size, frame rate and bandwidth usage levels.

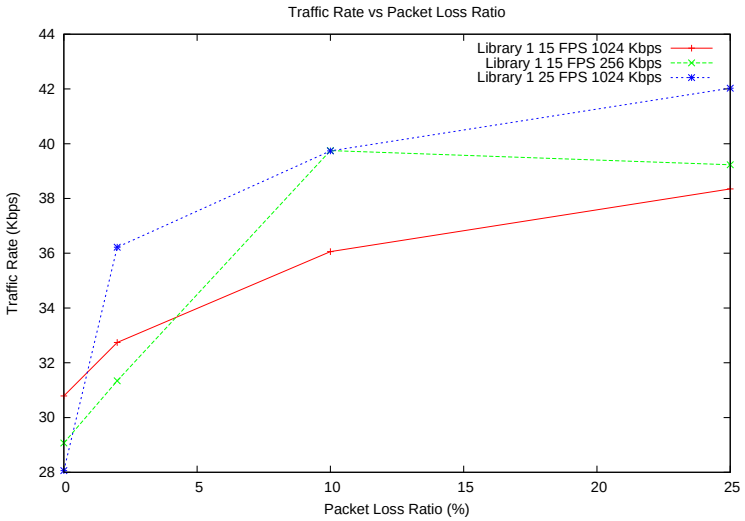


Figure 6.7: Traffic Rate vs. Packet Loss ratio without the effects of motion (Device 1)

Figure 6.7 shows the correlation of video source rate for three different configurations of Library 1, namely 15 FPS and 1024 Kbps, 15 FPS for 256 Kbps and at 25 FPS for 1024 Kbps. The video source rate was noted down for packet loss ratios such as 0%, 2%, 10% and 25%.

The graph shows that the video source rate clearly increases with increasing packet loss ratio. The increase in video source rates is different for different configurations but a clear trend is visible in these graphs. The source rate appears to decrease a bit for the case when the maximum allowable bit rate is 256 Kbps. But this may be because in order to achieve the low target bit rate, the video source rate is decreased. But it is still visible that the source rate at 25% packet loss ratio is still much greater than that at 0% or 2%.

It is visible that the increase in source rate is sharp for lower values of packet loss

ratio as compared to higher values. Thus it can be concluded that an aggressive source rate shaping strategy is employed for lower packet loss ratios than at higher ratios and that no further rate shaping takes place in lossier conditions.

In videos where random motion effects are present, the source rate may vary as can be seen in later results leading to absence of evidence pointing out to this trend. In the case of other configurations (e.g. frame rate of 5 FPS), the traffic rates were lower but the trend of increasing source rate in lossier conditions is evident. If the frame sizes are plotted for different configurations for different values of the packet loss ratio, it can be seen from Figure 6.8 that the frame sizes increase steeply for low values of packet loss ratio (less than 10%) and then increase smoothly for higher values. From the evidence, one of the explanations is that spatial or quality redundancy is added to the frame.

However, an alternative explanation for the increase in the average frame size can be that more I frames are sent by the source so that the decoder can repair the decoding context despite the loss of reference frames.

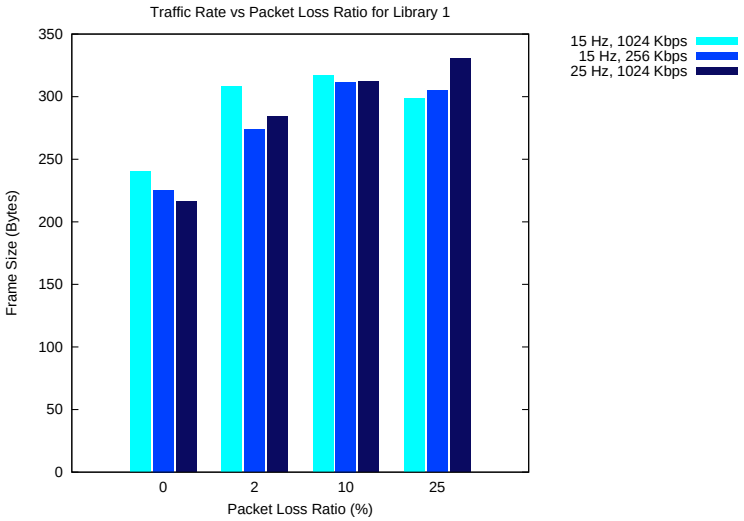


Figure 6.8: Frame Sizes vs. Packet Loss ratio without the effects of motion (Device 1)

In the absence of video with motion effects, DUT 1 was observed to code each frame in 2 packets. Similar results were seen in this experiment where DUT 1 coded each frame into 2 packets for conditions of no packet loss. However at higher packet loss ratios the average frame size increased to 2.05 packets per frame which meant that few 3 packet frames were detected.

As the loss of a packet may result in complete loss of a frame, the frame rate may be increased to ensure that despite the loss of some frames, video of substantial frame rate may be received by the receiver. This adds temporal redundancy to the video as compared to spatial redundancy discussed above. However, the frame rate did not vary substantially in face of packet loss which strongly suggested that Library 1 used spatial redundancy as a method of combating packet loss.

The measurement for frame sizes for DUT 2 appear to show a similar trend seen in Figure 6.9 where the frame sizes increase with the increase in lossy conditions. DUT 2

however has on an average higher data rate as compared to DUT 1 and hence in the case where the bit rate is limited to 256 Kbps, DUT 2 has to readjust frame sizes and reduce them in order to limit the data rate rate. This reduction may be due to reduction in resolution or video quality. This reduction allows DUT 2 maintain the low target bit rate as spatial of quality redundancy is added to the frame.

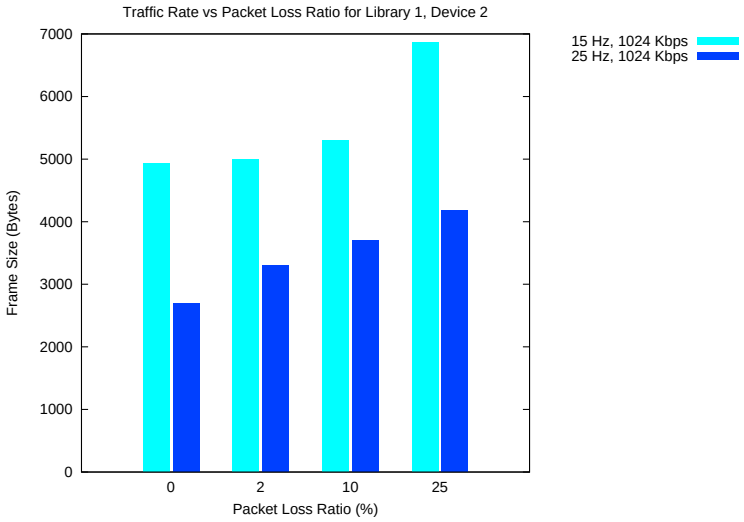


Figure 6.9: Frame Sizes vs. Packet Loss ratio without the effects of motion (Device 2)

The number of packets each frame is coded into is larger for DUT 2 as compared to DUT 1. This makes DUT 2 more susceptible to packet loss as compared to DUT 1 as then even the loss of 1 packet can render the entire frame useless. For lower values of packet loss conditions, each frame is encoded into an average of around 2.1 to 4.64 packets per frame (for different configurations) and for packet loss values up to 25% it increases to around 7.1 packets per frame. However frame rate remains nearly constant for changing packet loss ratios. This is conclusive evidence that spatial or quality redundancy and not temporal redundancy is added to each frame by Library 1 to counter lossy conditions.

If FEC is used as a method of error correction, parity packets may or may not be differentiated from media packets by using a different Payload Type number in RTP. As no packets carrying a different payload type number were observed, no evidence is observed that the library uses FEC for error correction.

6.3.1 Effect of Packet Loss on Video with Motion

Video with motion has a higher data rate as compared to video without motion. This causes an increase in frame rates and frame sizes leading to increase in number of packets per frame. As the number of packets each frame is coded into increases, the call becomes more susceptible to packet losses as each lost packet leads to a lost frame. The effects of motion on frame sizes and frame rates are highly unpredictable the evidence of which is seen later in this chapter and there are too many factors like the library trying to utilise the available bandwidth, limitations in frame rate due to configuration, target bit

rate and hence no substantial evidence of traffic rates being solely affected by packet loss ratio is seen.

No duplicate packets were observed and hence we conclude that the library does not use retransmission to combat packet loss. If the library used source rate control, a sharp decrease in the source rate would be observed. It has been seen that Skype uses source rate control to deal with packet loss [92]. As no substantial decrease in source rate was observed for any of the cases, it can be concluded that Library 1 does not use source rate control as a method of dealing with packet loss.

At lower frame rates (5 FPS), the frame sizes appear to increase suggesting addition of spatial redundancy. For higher frame rates (15 FPS), there is less processing time available per frame and hence the resolution is decreased so that the same number of frames can be encoded along with redundancy in the same amount of time.

For 2% packet loss, the deprecation in video quality was detectable but not obvious. For values of packet loss at 10%, serious deterioration in video quality was observed. The video appeared to be choppy and the amount of motion in it was considerably reduced. This can be attributes to loss of complete frames. As the loss of even one packet leads to loss of an entire frame, more number of frames are lost resulting in choppy video. In such conditions, audio video synchronisation loses meaning as lip movements cannot be deciphered. When the packet loss ratio was 25%, the video quality was bad and no communication was possible.

6.3.2 Effect of Packet Corruption

Adding packet corruption to the network shows similar results to packet loss. A corrupt packet is as good as a lost packet and hence error detection techniques are commonly used at the application layer to detect any corruption in data. Figure 6.10 shows that the source rates at higher values of packet corruption are higher than source rates at lower values of packet corruptions. The lower source rate(for 25 FPS, 1024 Kbps) at 2% value of packet corruption ratio is due to a lower frame rate which was observed at the source. This variation in frame rate can be considered an aberration as it has been seen that the frame rates tend to vary a lot for video conferencing applications.

It can be seen in figure 6.11 that again, spatial redundancy is being used to deal with corrupt packets. The frame sizes increase suggesting that more redundancy is packed into the frame as the library detects that some of the packets at the receiver have been corrupted. The number of I frames in the trace would also increase leading to an increase in the average frame size in the trace.

Google Hangouts

When this experiment was performed on Google Hangouts, no evidence of source rate shaping was observed. The source rate did not increase with a particular trend suggesting that FEC or similar error correction/lost packet recovery techniques were not used. The frame sizes for DUT 2 were observed to increase for increase in packet loss ratio. This suggests that spatial or quality redundancy might be added by Google Hangouts for DUT 2. But is is highly unlikely that different techniques for error correction will be used for different devices. A more likely explanation is that the number of I frames

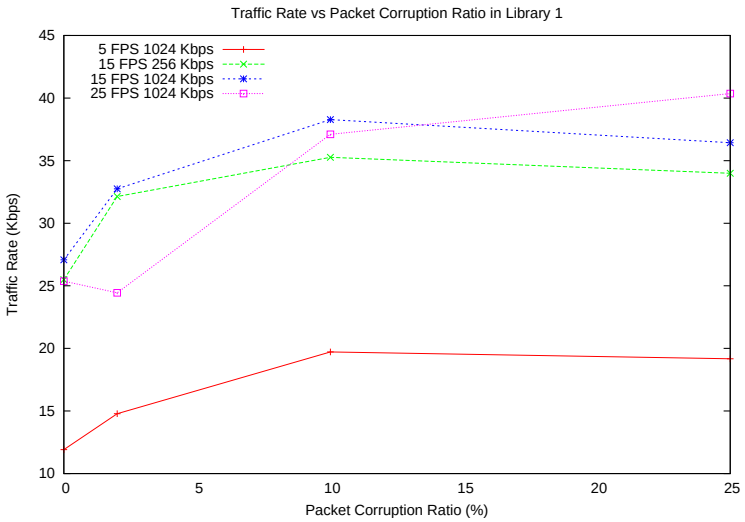


Figure 6.10: Traffic Rate vs. Packet Corruption ratio without the effects of motion (Device 2)

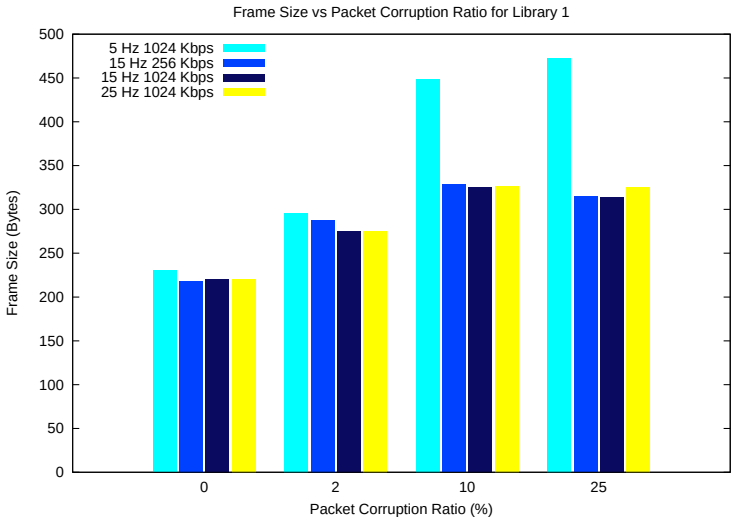


Figure 6.11: Frame Sizes vs. Packet Corruption ratio without the effects of motion (Device 1)

generated by the library increases to ensure maintenance of video decoding context in lossy network conditions.

Google Hangouts maintained a particularly high frame rate (around 30 FPS). At high frame rates, even the loss of half of the frames will provide a decent video quality at 15 FPS. This makes Google Hangouts more robust to packet loss and can provide a reasonable quality of experience. It has been determined from previous studies that Google Hangouts shapes video at the server and that each receiver gets a personalised video stream from the server. Our framework successfully detected that Google Hang-

outs recodes packets at the server. For each packet in the uploaded video stream, the framework tries to find the same packet in the download stream of the receiver. However as the same packet was not found, it is concluded that reshaping takes place at the server.

However this framework monitors video source rate, the frame rate, frame sizes and the bandwidth usage of the source device, any changes in the frame rate and frame sizes in the downloaded video are not calculated. However, the data is recorded and can be monitored to identify the frame rate and frame sizes of the received video. Some frames can be dropped to reduce the video bandwidth usage. By noting the difference in the upload and download frame rates, it can be found out how many frames were dropped.

Google Hangouts was known to use the scalable video coding approach previously. In a scalable video coding approach, each frame is coded as a number of layers where the one layer (the lowest) is sufficient to provide suitable video quality. However if the other layers too can be decoded in addition to this layer providing a better video quality. The server may send the receiver all the layers in order to enable the receiver to have a high quality video. However in lossy conditions, the server may drop some layers in order to reduce the source rate of the video. By observing the frame sizes of the received video as compared to the sent video, it can be determined whether the server drops some layers of the video before sending it to the receiver. But then Hangouts switched to VP8 codec where scalable coding approach was not used. Hence no further investigation into this approach was required.

6.4 Effect of Network Latency

This experiment investigates the effect of a large propagation delay on the performance of the library and also investigates whether the library tries to employ any techniques to maintain quality of experience.

To measure the effects of network latency, the library was configured at 25 FPS and 1024 Kbps (so that bandwidth usage would be high) and the delay was periodically increased for the following values: 100 ms, 200 ms, 400 ms, 1000 ms and 2000 ms. For values of delay larger than 2000 ms, the user is more than likely to drop the call and hence values more than 2000 ms were not considered.

Figure 6.12 plots video bandwidth for different cases for Library 1 and Google Hangouts. It can be observed that for Library 1, the bandwidth usage is fairly constant both when motion is present and is absent. This shows that no source rate shaping takes place for increasing values of delay.

For Google Hangouts, when motion is not present, the video bandwidth remains nearly constant even as the delay is increased. Video without motion uses less bandwidth as compared to video with motion. However for when the video had motion components, the video source rate progressively decreased for higher values of delay until it reached a low value beyond which it stopped decreasing the video source rate.

This can be explained as increase in the Round Trip Time (RTT) is interpreted as congestion caused by video overloading the available bandwidth. Hence the library decreases the source rate of the video to a point where video of sufficient quality can

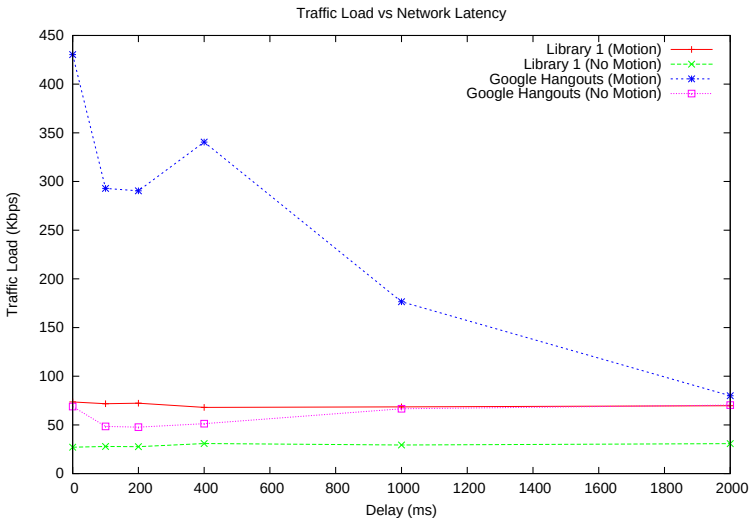


Figure 6.12: Bandwidth Usage vs. Network Delay

be delivered. This decrease in video may be caused by decrease in frame rate, video resolution or frame quality.

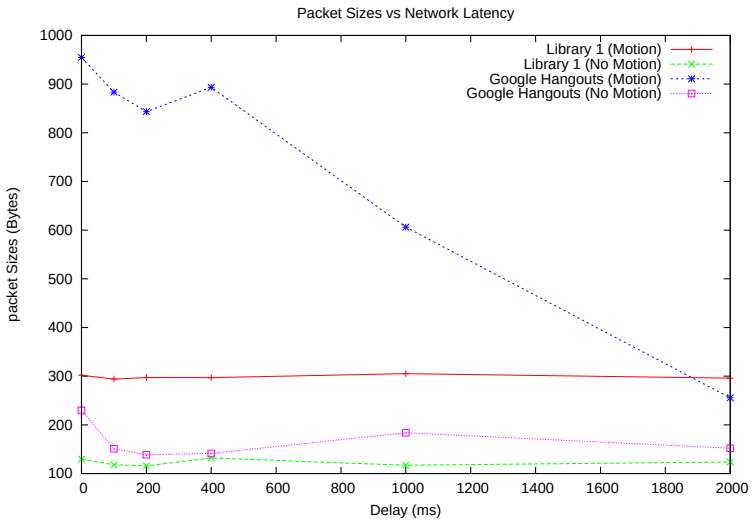


Figure 6.13: Frame Sizes vs. Network Delay

The decrease in the video source rate can be explained by observing the frame sizes in the video for different values of latency. The frame rate was nearly constant and hence the decrease in video source rate was due to decrease in frame sizes or frame quality. It can be observed that the packet sizes follow the same profile as the video source rate. The number of packets per frame was nearly constant in this case and hence larger packet sizes were indication of larger frame sizes. This suggests that the decrease in source rate for Google Hangouts was due to reduction in video resolution or quality.

For Library 1, the disconnection time (the time at which the client disconnects from the server due to a large network latency) was observed to be as high as 9000 ms. This is however very tolerable as the user is more likely to disconnect the call himself if and when the delay exceeds 2000 to 3000 ms.

6.5 Effect of Bandwidth Variation

This experiment studies video adaptation for both Library 1 and Google Hangouts under different network conditions. It observes how the libraries adjust to different bandwidths and how do they ramp up its bandwidth usage in case of good network conditions.

Different configurations of Library 1 use different bandwidth.To observe properly the effects of changing bandwidth conditions, it was necessary that the library was actually allowed to use maximum bandwidth available and then see if it can still adapt to lesser bandwidth conditions. Library 1 was configured at 30 FPS, the maximum allowable bit rate was set at 1024 Kbps and the capture resolution was set to 800 X 600.

Another goal of this experiment is also to determine how fast the library can ramp up its bandwidth in the case of abundant bandwidth and whether it can detect low bandwidth conditions and adapt its initial sending rate accordingly. The configuration parameters specify the upper limit at which the video will be encoded and the video may never exceed these limitations although a lower quality video may be streamed by the library depending on various factors.

To simulate low quality connections, the bandwidth was fixed to 64 Kbps. The experiment was repeated for bandwidth limitations for 128 Kbps, 256 Kbps, 512 Kbps and the full available bandwidth which was about 10 Mbps. A limit of 1 Mbps was not chosen as the video almost never exceeded 1 Mbps.

As one of the goals of this framework is to understand performance of Library 1, for this experiment, Library 1 was run on both DUT 1 and DUT 2 as conference initiators. This means that two instances of the experiment were run where in the first one only DUT 1 is connected to the framework and DUT 2 joins the conference from outside the framework. In the second instance, their positions are interchanged and performance of Library 1 on DUT 2 is determined. Analysis with DUT 2 helps us gauge the performance of the library for heterogeneity of hardware. For Google Hangouts, only DUT 1 was used as the conference initiator.

Video parameters like bandwidth usage, frame size and frame rate were measured at the interval of every 10 seconds for a period of 10 minutes so that more granularity can be obtained in the changes in packets and traffic rates. Then for additional analysis, a moving average of data rates, frame rates and frame sizes was computed in order to better understand variation in bandwidth usage, frame sizes and frame rates for a period of 600 seconds.

6.5.1 Full Bandwidth

Initially, no constraints were placed on the bandwidth to see how the library ramps up its bandwidth usage and tries to utilise better bandwidth. The library was monitored for a period of 600 seconds and the sampling interval (in order to generate periodic data)

was 10 seconds. This allowed us to minutely observe changes in frame size, frame rate and bandwidth usage over each period of 10 seconds.

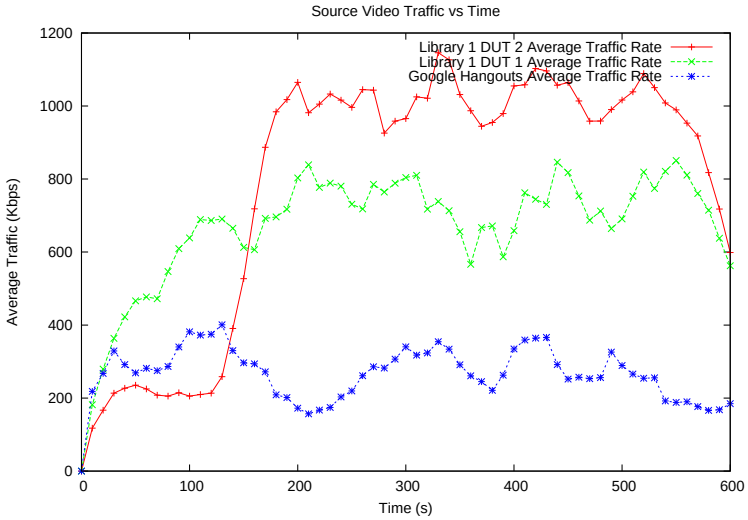


Figure 6.14: Source traffic rate Library 1 on DUT 1 and DUT 2 and of Google Hangouts on DUT 1

Figure 6.14 shows that for DUT 2 Library 1 uses less bandwidth at the start of the call and ramps up its bandwidth usage 120 seconds into the call to rapidly increase the traffic rate to ensure high bandwidth utilisation. On DUT 1, for the same configuration, the bandwidth usage ramps up smoothly as compared to on DUT 2. Moreover, it can be seen that the peak bandwidth usage for DUT 2 is much higher than that for DUT 1.

If traffic rates for Library 1 and Google Hangouts for DUT 1 are observed, it can be clearly seen that Library 1 starts at a similar traffic rate as compared to Google Hangouts but starts ramping up the bandwidth usage 30 seconds into the call. It can also be seen that the bandwidth usage for Library 1 on an average is much higher as compared to Google Hangouts. The average traffic load has been computed by taking a moving average of the traffic load computed every 10 seconds by the framework. It can be observed that Library 1 has been observed to use up to 1305 Kbps bandwidth for a call on DUT 2 and 971 Kbps for a call on DUT 1. Meanwhile the maximum bandwidth usage for Google Hangouts was 558 Kbps.

The actual traffic load oscillates heavily due to the presence of jitter. Jitter is the variation of the network latency in time. At low values of delay, the rate of packets reaching the receiver might be much higher than the rate at which packets are consumed by the receiver. However at high values of delay, the rate of packets reaching the receiver may not be high and there is a threat of the buffer being rendered empty by the fast consuming receiver. In the case of network based video applications like video on demand, this jitter can be tolerable as the video is already encoded and hence the source rate can be varied by temporal or spatial scaling and in the worst case, the receiver can interrupt the video rendering as it waits for new packets to arrive, a process which is commonly known as buffering. Video conferencing, being a real-time interactive application, has

to have very small values of buffering delay (a larger de jitter buffer increases delay) and hence has less tolerance to jitter. Hence video conferencing applications have to vary their source rate appropriately in order to ensure interactivity amongst users by managing the jitter in the network.

Frame Rates

The source rate can be varied by varying the frame rate or frame size. Figure 6.15 plots the frame rates measured every 10 second by the framework. It can be observed that Google Hangout maintains a relatively higher frame rate as compared to Library 1. Hangout starts at a frame rate of around 27 Hz and maintains a frame rate in excess of 25 Hz for most part of the call duration. For Hangouts, a highest frame rate of 30.4 Hz has been observed and a lowest frame rate of 10.8 was seen. Library 1 however uses low frame rates as compared to Google Hangouts. In the case of Library 1, frame rate as high as 15.6 Hz and as low as 5.5 Hz was observed for DUT 1 and for DUT 2, a frame rate as high as 14.5 Hz and as low as 2.5 Hz were observed. A video of higher frame rate is more robust to packet loss as the even the loss of some of the frames would still lead to a decent frame rate. Higher frame rate captures motion more efficiently and increases the quality of experience.

The oscillation in frame rates is due to the variation in the motion content and the level of detail captured in each frame apart from the need to constantly change the source rate. When the frame rate is higher, the video has more amount of motion and more artefacts as the hardware may have less time to compress each frame. Various de-blocking and de-ringing filters are used by the codec to iron out the artefacts and these filters require high CPU usage. Thus high frame rates may overload the CPU resulting in lossier compression resulting in lossier compression. Apart from that, events like OS glitches can produce a minor variation in frame rates.

However, it has been discussed in this section how the source rates vary and one of the cause of these varying source rates is the varying frame rate. Figure 6.15 confirms this by plotting the frame rate. However, the frame rate is not the only criteria which determines the quality of experience. A library may encode 30 frames in the first second and only 4 in the other leasing to an average frame rate of 17 Hz. But the change in the the frame rate causes choppy video and greatly reduces quality of experience. Linearity of frame rate is determined by taking a moving average of the frame rate and it helps us understand the variation in frame rate better. From the linearity plots, it can be seen that Google Hangouts maintains a much higher frame rate than Library 1. Library 1 maintains a slightly higher frame rate for DUT 1 as compared to DUT 2 as DUT 1 has more powerful hardware.

However the linearity curve for Library 1 is smoother as compared to Google Hangouts meaning that the frame rate for Library 1 tends to vary less as compared to Google Hangouts. However the average frame rate in Hangouts always remains between 20 and 25 Hz and the slight variation in the frame rate may not be visible to the viewer at higher frame rates. In the case of library 1, the variation in frame rates, even though much less as compared to Hangouts, can be detected by the user as it falls below 10 Hz. Below 10 Hz, video tends to be choppy and as lip sync cannot be seen clearly, audio video

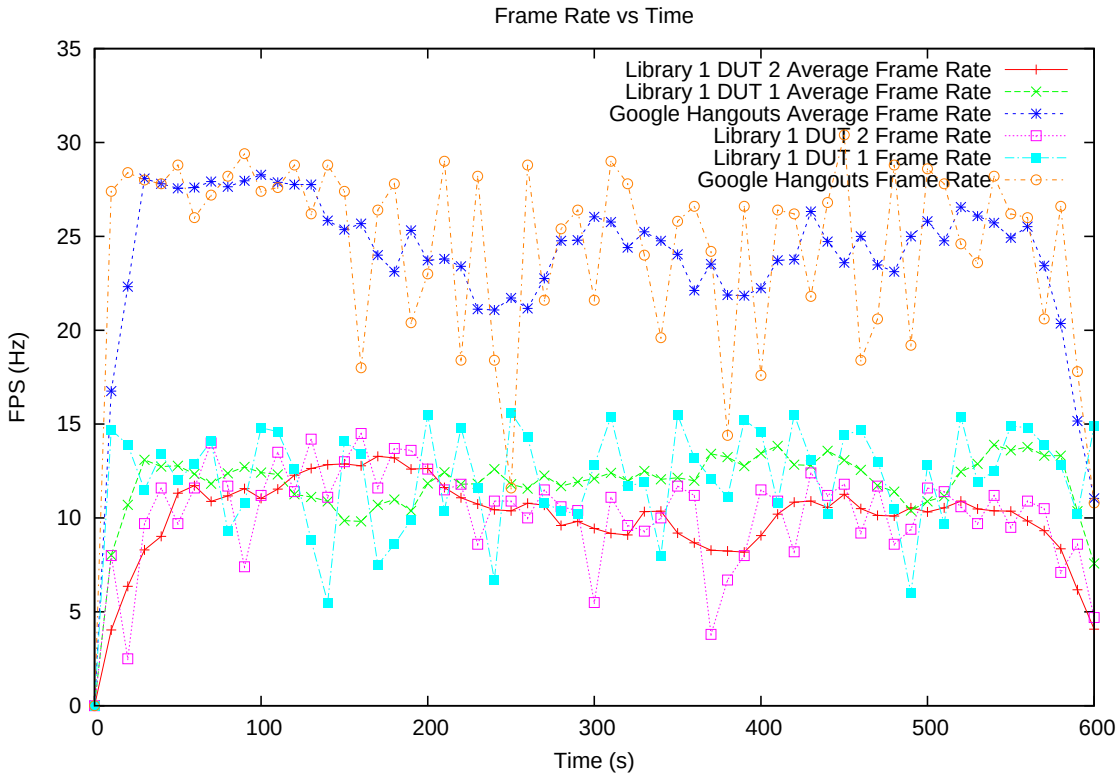


Figure 6.15: Frame rate and Linearity of Library 1 on DUT 1 and DUT 2 and of Google Hangouts on DUT 1

synchronisation loses meaning. However for thumbnail videos with less motion, the low frame rates might be sufficient at providing the user with a good quality of video.

Frame Sizes

Video quality cannot be gauged by frame rates alone. A video with a high frame resolution and quality at a frame rate of 15 Hz and a video with comparatively low frame quality and resolution but at a significantly higher frame rate (30 Hz) may both be considered good by viewers. Hence the frame sizes of Google Hangouts for DUT 1 and Library 1 for both DUT 1 and 2 are plotted to observe the change in frame sizes.

It is observed from Figure 6.16 that the average frame size is much higher for library 1 as opposed to Hangouts. This higher size indicated higher quality or higher resolution. This higher size is the reason for higher traffic rates. Library 1 starts with a lower frame size and increases the resolution and reaches nearly the full resolution at around 90 seconds into the call. Hangouts shows more variation in frame sizes as compared to Library 1 but on an average has much lower frame size. The frame size for DUT 2 for Library 1 is much higher as compared to DUT 1. This might be because of the fact that DUT 1 has more powerful hardware as compared to DUT 2. DUT 2 uses low frame sizes initially and once it detects that sufficient bandwidth is available, it ramps up the frame

sizes. This however leads to increases bandwidth usage and low robustness in conditions of varying bandwidth.

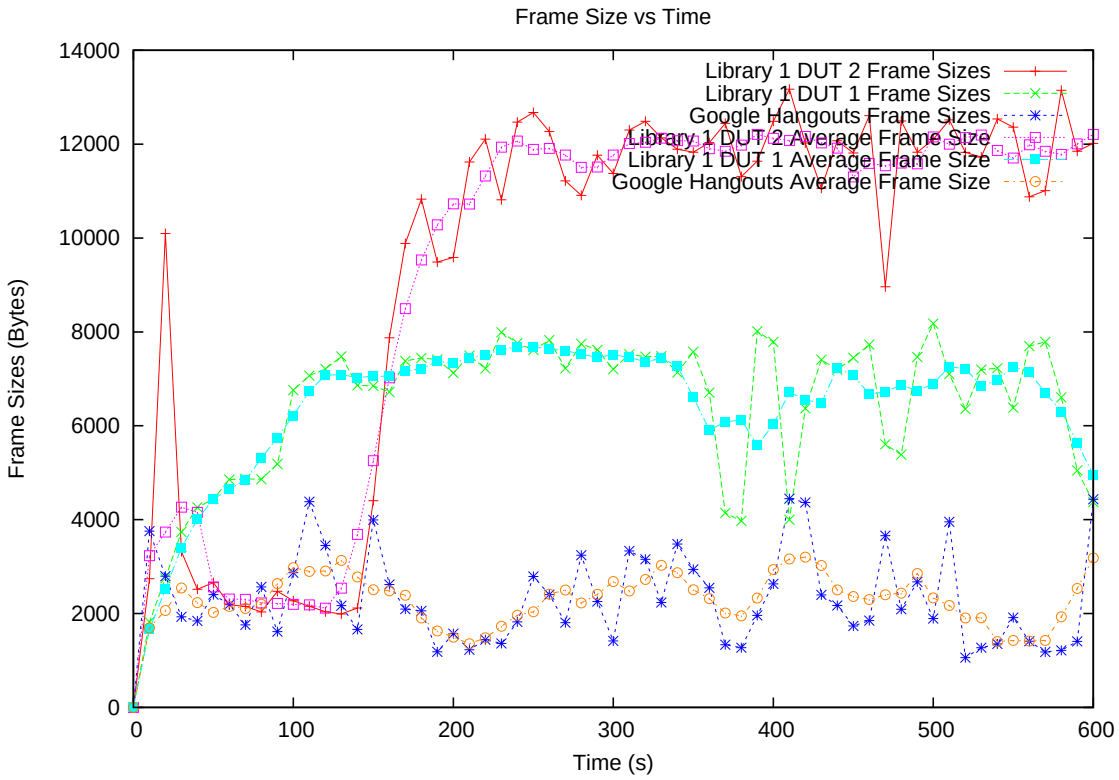


Figure 6.16: Frame Size and Average Frame Size of both the libraries when full Bandwidth is available

The framework detected that Google Hangouts re-encodes each frame at the server. This re-encoding delay increases the source to receiver delay. Due to higher delay, Google Hangouts is more susceptible to jitter and congestion and has to vary the source rate to ensure robust performance. Lower frame sizes reduce the processing complexity at the server thus reducing the delay.

6.5.2 Performance in Low Bandwidth Conditions

When a video conferencing library suddenly has to face low bandwidth conditions while operating under high bandwidth conditions, the video adaptation to a lower bandwidth usually takes some time and meanwhile the RTT to the server increases and reaches an extent where the connection breaks and the call drops. Hence low bandwidth conditions were created before the call was established and then the performance of the call was tracked. For low bandwidth conditions, two bandwidth options were selected, 64Kbps and 128 Kbps.

Bandwidth limit 64 Kbps

For the bandwidth limit of 64 Kbps None of the libraries was able to sustain a call for more than 120 seconds. Library 1 started with a traffic source rate of more than 180 Kbps both in the case of DUT 1 and DUT 2. This caused a very high RTT to the server and the call dropped immediately in less than 20 seconds. Hangouts were comparatively more robust and a call was sustained for around 100 seconds. However the traffic source rate increased above 64 Kbps causing the RTT to increase and dropping the call.

Google Hangout started with a low frame rate and maintained the frame rate for around 90 seconds. After that it increased the frame rate while decreasing the frame sizes by a quarter. However the readjustment was not enough to decrease the source rate and the call dropped due to increase in RTT from video overload. The frame rate increases starts around 5 FPS and increases up to 20 when the source rate increases beyond the threshold causing disconnection. Library 1 starts with very high frame sizes which causes increase in RTT due to video overload.

Bandwidth limit 128 Kbps

For a limit of 128 Kbps, Google Hangout was robust enough to sustain a call whereas Library 1 failed to sustain a call for more than 30 seconds. Google Hangout is able to sustain the average traffic load to a value way less than the available bandwidth and also manages to keep the peak traffic load also less than the available bandwidth. Figure 6.17 plots the video source rates for Google Hangouts for bandwidth conditions of 128 Kbps, 256 Kbps and 512 Kbps. The video source rate for Library 1 was plotted only for DUT 1 for available bandwidth of 512 Kbps and DUT 2 for the available bandwidth of 256 Kbps as in all other cases, the bandwidth usage was high leading to a call drop and call times in excess of 120 seconds were not observed.

6.5.3 Performance in Medium Bandwidth Conditions

Under medium bandwidth conditions, the bandwidth limit was fixed to 256 Kbps and 512 Kbps.

Bandwidth limit 256 Kbps

At 256 Kbps, it was observed that Library 1 starts with a traffic source rate of just under the available bandwidth while Google Hangouts starts with a rate much less than 100 Kbps. Hangouts then ramps up its available bandwidth and reaches an average bandwidth near the available bandwidth while ensuring that the peak traffic rate does not exceed the available bandwidth. Library 1 starts with source rate just less than 256 Kbps and ramps it up to more than 500 Kbps. The RTT increases due to video overload and the call drops. However in the case of DUT 2, Library 1 keeps its source rate below 256 Kbps.

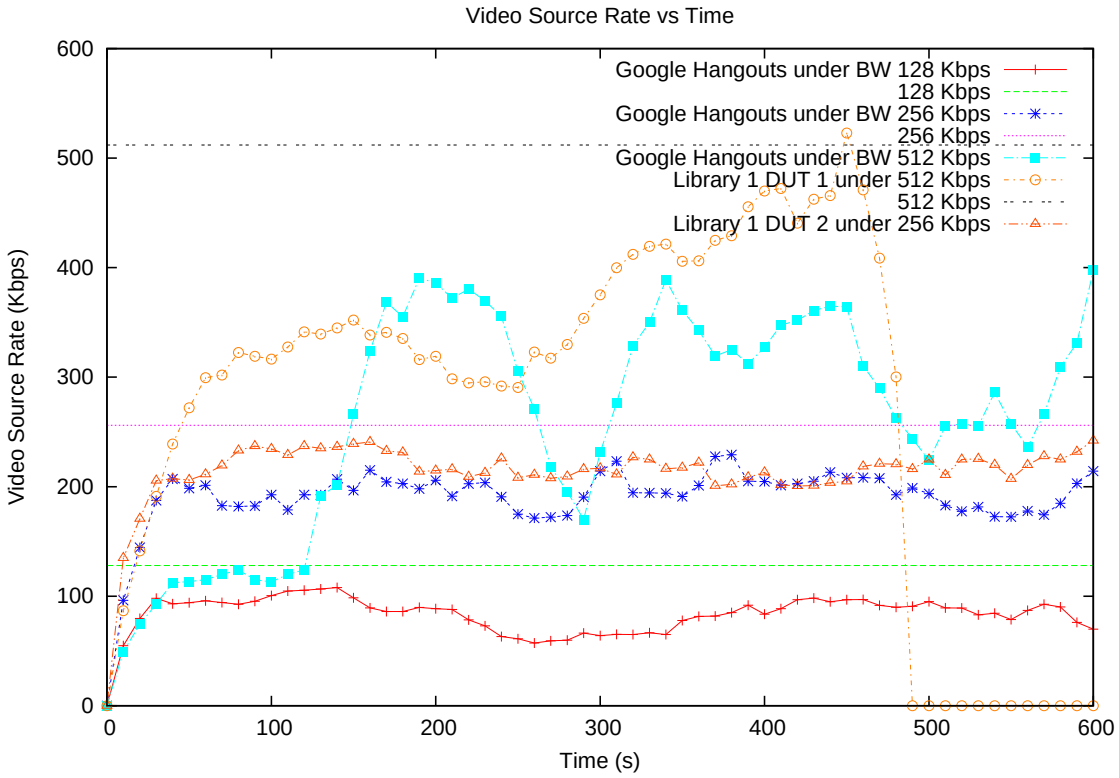


Figure 6.17: Video Source Rate of Google Hangout and Library 1 under different bandwidth conditions

Bandwidth limit 512 Kbps

At 512 Kbps, Google Hangouts was able to maintain a source rate well below 512 Kbps which ensured a robust call. Library 1 however held a call on DUT 1 for around 8 minutes before the source rate increased causing runaway increase in RTT and disconnection. This behaviour can be observed from Figure 6.17.

The video bandwidth used by a Hangout call was always observed to be considerably less than the available bandwidth. This bandwidth is utilised for audio calls, signalling, other application level data sharing (text) and other flows emanating from the same machine (competing TCP and UDP flows). This bandwidth gap also allows for robustness in case of jitter as there is enough margin to vary the source rate.

For Library 1, the call was not sensitive to bandwidth limitations and the in most of the cases dropped due to video overload. In some cases, the library manages to sustain a call beyond 300 seconds but in those cases, the average source rate occupied a greater portion of the bandwidth. This results in less available bandwidth for audio, signalling, other application level data and TCP/UDP flows from the same machine. From this we concur that Library 1 is less sensitive to bandwidth limitations and must be carefully configured in order to prevent overload of the bandwidth.

The bandwidth sensitivity of Google Hangouts is further investigated by observing

frame rates and frame sizes throughout the duration of the call.

Frame Rate of Google Hangouts The frame rates of Google Hangouts were observed for different values of available bandwidth. Figure 6.18 plots the frame rates for Google Hangout for different bandwidth conditions. Google Hangouts tends to maintain a frame rate of between 20 and 25 Hz for a call. However when the available bandwidth was 128 Kbps, for some periods of time, the average frame rate dropped to as low as 10 Hz during the call. Thus it can be concurred that Google Hangouts places more emphasis on maintaining a high frame rate and that QoE of a call under low bandwidth conditions would be less due to varying frame rate. However, as frame rates are still higher than 10 Hz, audio video synchronisation is possible (lip sync corresponding to audio can be identified).

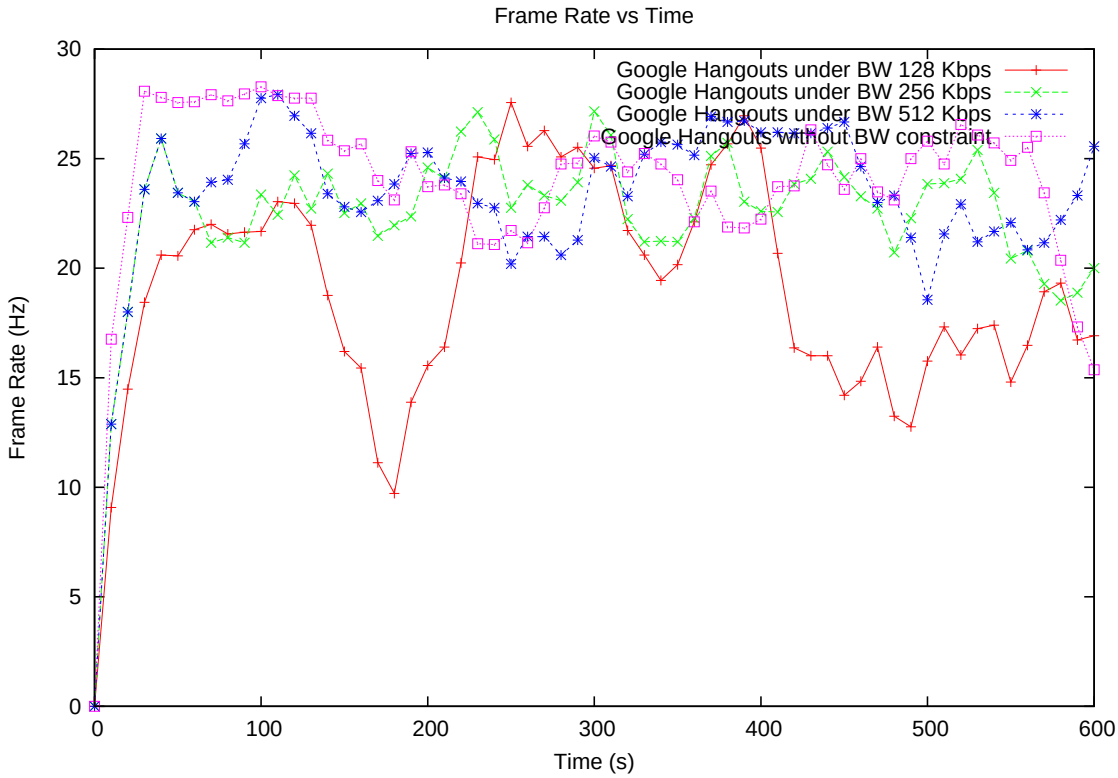


Figure 6.18: Frame rates for Google Hangout under different bandwidth constraints

As discussed earlier, frame rates in combination with frame sizes determine the call quality. The trend in frame sizes and frame rates with traffic rate is consistent under 512 Kbps bandwidth restrictions. Google Hangouts has a source rate well under the maximum bandwidth and uses occasionally the peak source rate increases but never increases beyond the available bandwidth. Library 1 is able to keep its source rate below the available bandwidth but once it detects that the bandwidth is not being utilised fully, it increases the source rate to a point where it increases beyond the available bandwidth causing runaway increase in the RTT leading to call drop.

The frame sizes are plotted in Figure 6.19 and it can be observed Google Hangouts uses low sized frames (meaning low quality and low resolution) to keep video source rates low in low bandwidth conditions. However the frame sizes vary a lot through out the duration of the call which means that the user sees high quality video at some points and low quality at some other points. The frame sizes increase when the available bandwidth is 256 Kbps and still increase when the available bandwidth increases. This suggests that Google Hangouts primarily uses scaling in frame quality/resolution while maintaining a high frame rate as the method of varying source rate. The variations in source rate at low bandwidth are necessary as the margin for handling jitter reduces at low bandwidths. When higher bandwidth is available, Google Hangouts delivers better quality video. However the high frame rate enables Google Hangouts maintain a decent QoE even when the video has low resolution/quality.

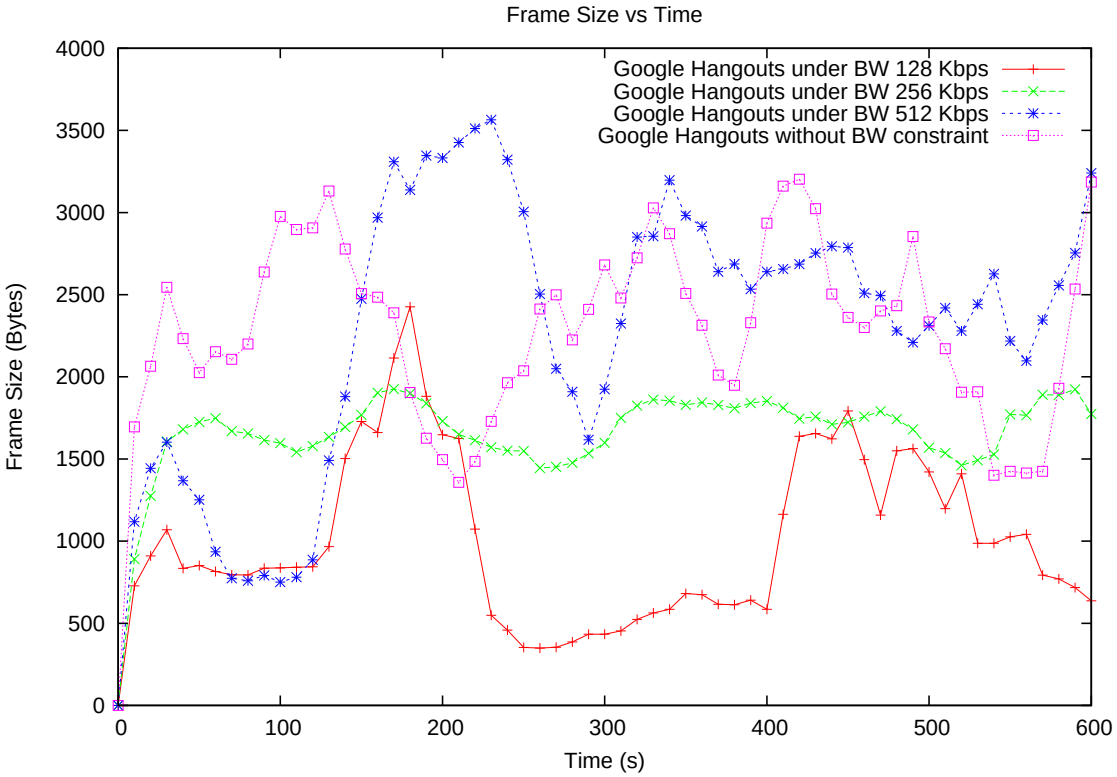


Figure 6.19: Frame sizes for Google Hangout under different bandwidth constraints

Failure Analysis

The results of this experience can be used to determine the cause of failure of a library in the case of less available bandwidth. The call session fails as the video overloads the bandwidth. This causes runaway increase in RTT leading to excessive delays. As we have discussed earlier, Video source rate can be reduced by reducing either the frame sizes or frame rates. Thus frame sizes and frame rates are observed to pinpoint the cause

of the failure.

The failure of library 1 to sustain a call on DUT 1 under available bandwidth of 512 Kbps is used as a case study here determine the cause of failure and suggest methods to ensure that the call does not fail. Library 1 maintains an average frame rate between 10 and 15 Hz throughout the call. Frame sizes at the beginning of the call are observed to be around 1800 bytes which then later increase. At around 480 seconds, the video bandwidth exceeds the available bandwidth causing a failure. Hence the video source rate must be maintained below the available bandwidth at all point of time during the call.

Video bandwidth should be ideally 80% of the total available bandwidth. The remaining bandwidth is used for audio traffic, signalling traffic and other competing TCP and UDP flows from the machine. Hence the video bandwidth usage should be around 410 Kbps for a sustainable call. The video bandwidth can be controlled by controlling the frame rate or the frame sizes. Frame rate cannot be decreased further below 10 Hz as they would result in choppy video and identifying lip sync will not be possible. Hence the frame sizes should be limited to limit the video bandwidth usage.

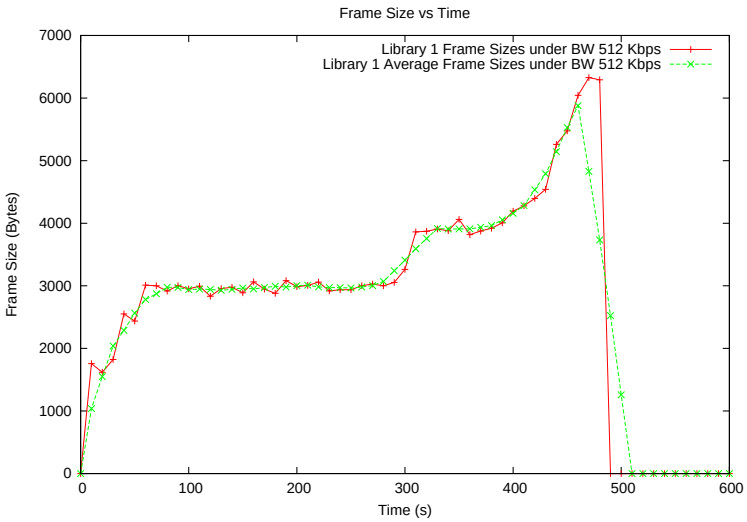
It can be observed from Figure 6.17 that the video achieves a target bandwidth usage of around 400 Kbps in the time period between 300 and 400 seconds. The frame sizes and the average RTT to the server between that time period can be seen in Figure 6.20. It has been suggested that the ideal value of RTT is around 150 ms [38]. It can be corroborated that the RTT is around 150 ms in the time period between 300 and 400 seconds into the call. The figure shows that as the frame sizes keep on increasing, the RTT increases due to the video overloading the network.

The library should be tested for different configurations using the framework to check which configuration achieves the target frame sizes while maintaining an RTT to the server of about 150 ms or lower. That particular configuration can be used for the target bandwidth. If runtime, reconfiguration of video bitstream is available, that particular configuration should be used when it is detected that the available bandwidth is 512 Kbps.

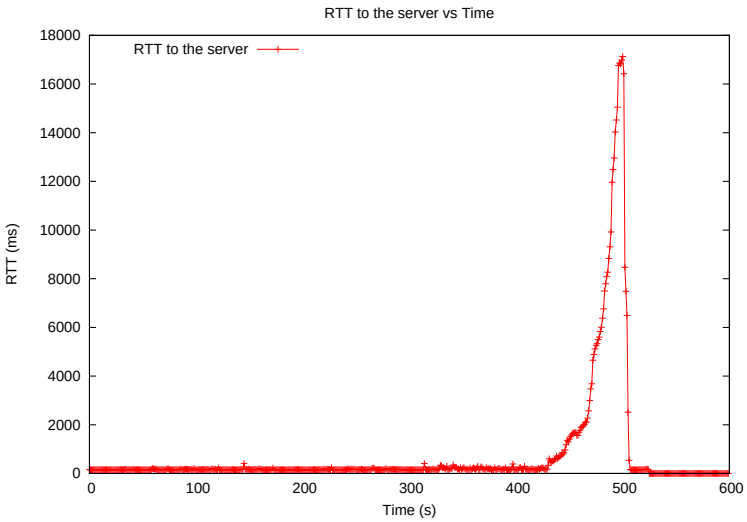
6.6 Pre-Connection Results

The framework is designed using Wireshark and a simple observation of the Wireshark traces reveals a lot of useful information about behaviour of the library. Apart from the media streams, the traces also reveal information about the pre-connection process of the library. A pre-connection process is defined as the actions a library takes from the moment the call is initiated by the user to the point until the first media packet is seen. This information is relatively easy to obtain and can be included in the framework in the future.

The pre-connection procedure of Library 1 was observed to find out the complications and the time required before the first media packet was sent. This information can be used at the application design level to determine when a connection request is to be made in case seamless connectivity has to be provided to the user of the application. The login process of library 1 can be best described as a three step process as seen in Figure 6.21.



(a) Frame Sizes of Library 1 for DUT 1



(b) RTT to the server

Figure 6.20: Increase in Frame Sizes accompanied by increase in RTT due to video overdrive

1. The first step is carried out when the application is launched by the user. The application performs a handshake with a server meant for verifying whether the user is a valid user or not.

2. In the second step the library performs a handshake with another server which returns the IP address of a media relay server (and the port number) which will be used by the library for the session. Usage of different media relay servers (identified by different IP addresses in different subnets) has been observed.

3. In the third step, the media relay server performs a handshake with the client and

after identification and verification, the video stream starts relaying video data.

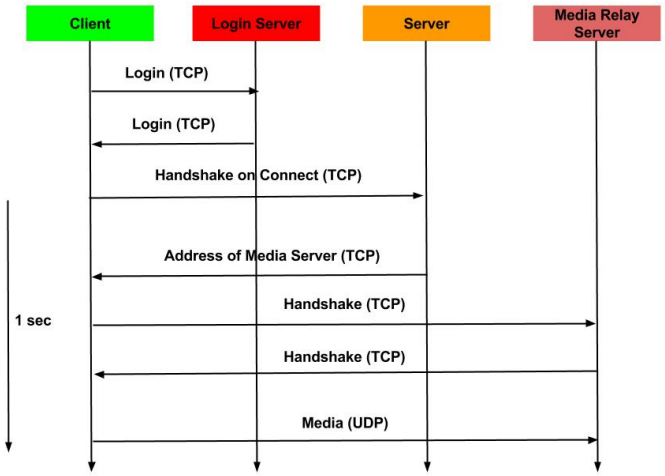


Figure 6.21: Login Process of Library 1

The system time stamp of the first packet sent on initialising the call was noted and the time stamp of the first media packet was noted. The average difference in the time stamps was found to be around 1 second. In conditions of delay of around 1000ms, the login process has been observed to take up to 7 seconds on account of multiple retransmissions. This information can be useful at the application design level where a request to initiate a call can be taken as early as possible depending on the needs of the user and the network conditions. Transmission Control Protocol is used throughout the login process and once login is established, the media packets are sent using UDP. Decoding the media stream using RTP results in a detailed analysis where port numbers, source identification numbers, sequence numbers and frame start boundaries can be identified.

7 Conclusions and Evaluations

Measuring qualitative multimedia parameters using quantitative analysis requires extrapolation. There are standards for video conference framework defined at various levels, but no standards to define performance and quality of experience. Quality of Experience (QoE) is a subjective parameter and may vary for each user. A video with a frame rate of 15 Hz at a high resolution may be perceived as 'good' as well as a video with a high frame rate and a low resolution. Hence it is difficult to predict the QoE based on measurement of QoS parameters like audio and video delay, frame sizes, frame rates and video source rates. However, by monitoring these variables, a near accurate insight into the behaviour of the library can be obtained. This information can then be used to determine the QoE up to a certain extent and the values of the variables can be determined for optimum performance. The framework discussed in previous chapters attempts to determine the variable values allowing the developer to extract information for optimum performance of the library through reproducible experiments. This information allows him to determine the robustness of the library under varying network conditions and helps determine major bottlenecks in offering a decent QoE.

The previous chapter discusses the results obtained after applying the framework on Library 1 and Google Hangouts. The conclusions derived from these experiments are discussed in the following sections.

7.1 Congestion - The main challenge to QoS

The internet is a best effort delivery network and hence the onus of meeting certain QoS (Quality of Service) standards lies on the application (or protocol) being used. Media applications use UDP which does not provide QoS at the transport layer and therefore the higher level application has to take various measures to guarantee the QoS. The main challenge to QoS is network congestion. Network congestion can present itself in the following ways:

1. **Delay** - Larger traffic means larger queues at routers leading to increased delay. Variable traffic on the internet (as is the case) results in varying amounts of queueing sizes leading to variable delay. This variable delay is called *Jitter*.
2. **Packet Loss** - In a network with high traffic, incoming packets may tend to saturate the queueing buffer leading to excess packets being dropped. This causes packet loss.

Apart from that if the available bandwidth is less than the data rate generated by the application, the data overload may cause congestion which can manifest itself in one of the two ways mentioned above. Hence any video conferencing library which aims to

provide a decent QoE should have appropriate congestion control and error control. To evaluate that, some questions were asked by the framework to each library which can be summarised as follows:

7.1.1 How does each Library react to Packet Loss/Corruption ?

Library 1

Library 1 responded to packet loss by increasing the source rate of the video. This increase in source rate was due to increase in frame sizes. Frame sizes increase as redundancy is added to each frame. It is not clear whether the added redundancy is due to usage of error correction techniques like FEC (Forward Error Correction) or due to simple redundancy in video resolution/quality as no information on error detection/correction/concealment strategies employed by the library is available. This is useful as if one of the redundant packets is lost, the other can be used to decode the frame without comparable loss in video quality. The average frame size can also be increased due to the presence of more number of I frames. The receiver loses decoding context due to packet loss and requests more I frames to repair the decoding context. I frames generate more data as compared to P frames leading to an increase in the average size.

This suggests that Library 1 is less error tolerant and must be deployed in situations where the available network conditions are good.

Google Hangouts

Google Hangouts adjusted to increased packet loss with a mild increase in the source rate. Hangouts uses a high frame rate which means that despite the loss of few frames, the frame rate is still enough to provide a decent quality of experience. The increase in source rate is due to the increase in frame sizes meaning that some redundancy is added to the video frame (or more number of I frames are detected in the trace). However in hangouts, the video is recoded at the server and hence it is not possible to determine how exactly the recoding affects the video.

However it can be observed that Google Hangouts is able to provide a decent video quality at values of packet loss ratio up to 10% as compared to Library 1 where the video quality deteriorates even at low values of packet loss ratio.

7.1.2 How does each Library react to Network Latency ?

Library 1

Performance of Library 1 is largely unhindered by increasing network latency. There is no noticeable change in the video source rate, frame sizes and frame rates. This suggests that Library 1 does not use total delay as a measure of congestion and good video quality may be delivered by the library even if the propagation delay between the end points is too high. Thus users may be present in different corners of the globe (leading to a large scale propagation delay) and still be able to enjoy a high quality video conference. This indicates that Library 1 favours video quality over interactivity.

Google Hangouts

It can be observed that Google Hangouts perceives the delay as an indicator of congestion and decreases the video source rate. The reduction in source rate is due to reduction in packet (frame) sizes and the frame rate is not varied significantly. This shows that Google Hangouts favours interactivity over video quality and tries to decrease the video quality in order to improve interactivity.

7.1.3 How does each Library react to Bandwidth Constraints ?

Library 1

Library 1 failed to sustain a call indefinitely when the available bandwidth was less than 1 Mbps. It did sustain a call for around 8 minutes when the available bandwidth was 512 Kbps but it attempted to increase the video quality leading to increase in video source rate. This overloaded the network with video leading to disconnection. From this we can conclude that Library 1 has no bandwidth detection system. The library tries to achieve the configuration suggested by the developer without monitoring the available bandwidth (which means, it decides that the developer knows what he is doing).

Library 1 maintained a low frame rate (despite the available bandwidth) of around 15 Hz even when the configuration allowed it to go up to 30 Hz. Frame sizes are frame quality were used to vary the source rate ensuring that the user got the highest possible video quality. This caused disconnection as soon as the source rate exceeded the available bandwidth. As the source rate increased, an increase in the round trip time (RTT) to the server was seen and the call was disconnected as soon as the RTT exceeded 9000 ms (which was earlier reported to be the maximum RTT at which call was still operational). This suggests that Library 1 monitors RTT to the server in order to decide whether to drop the call or not.

Ideally, video bandwidth usage should be 80% of the available bandwidth in order to account for audio, signalling and other competing TCP flows emanating out of the same machine. This suggested that Library 1 did not have good bandwidth awareness.

Google Hangouts

Google Hangouts uses a higher frame rate of between 25 and 30 Hz and considerably lower frame sizes. Google Hangouts favours motion over relative video quality. The higher frame rate allows Google Hangouts to be more robust to random packet loss (decent video quality can be offered at the loss of few frames).

When the available bandwidth increases, Google Hangouts increases its frame sizes (similar to Library 1), by either increasing video resolution or video quality to increase the video quality. However hangouts monitors the available bandwidth and always keeps its source rate below the available bandwidth. The video bandwidth usage for hangouts was in fact considerably less than the available bandwidth. This remaining bandwidth is required to ensure that good quality audio can accompany the video call. This gap in bandwidth (which can be referred to as the audio gap) suggests that hangouts gives higher priority to audio than video. This helps to deliver a better QoE as good quality audio can still sustain an interactive call in the absence of video.

It was observed that Hangouts does not saturate the link with audio/video as the highest bandwidth used by a Google Hangouts call was reported to be around 400 Kbps (even when the available bandwidth was in excess of 10 Mbps). Thus Hangouts does not use bandwidth efficiently and does not try to increase video quality for the user (even when higher bandwidth is available).

Apart from that, different configurations of the library were investigated for different call sizes in order to determine the effect of heterogeneity of the hardware and varying call sizes. The conclusions from those experiments are presented in the following sections.

7.2 Effect of Heterogeneity in Hardware

The experiments discussed in the previous chapter were performed on two devices, Device Under Test 1 and 2. DUT 1 was a Nexus 10 and DUT 2 was a Nexus 7. One of the main challenges in providing a good QoE over a multi-user video conference is the heterogeneity of hardware. DUT 1 was found to be more powerful as compared to DUT 2. Both Library 1 and Google Hangouts show better video quality on DUT 1 as compared to DUT 2.

DUT 1 used bandwidth far more efficiently as compared to DUT 2 without any noticeable loss in video quality. Also frame sizes observed for DUT 1 were far lower as compared to DUT 2. DUT 1 could compress the bit rate to a value of 256 Kbps (or a value lower than that) even if the target bit rate allowed by the developer was high. This suggests that DUT 1 can achieve higher compression as compared to DUT 2. DUT 1 is powered by a Samsung Exynos 5 chipset which is far more powerful than the NVIDIA Tegra 3 chipset which is used in DUT 2. This heterogeneity in hardware causes different bandwidth usage and different video quality. Higher bandwidth usage also places a limit on the call size. Hence if DUT 2 has to be used, the library has to be configured carefully for lower video quality to ensure that it uses less bandwidth.

7.3 Effect of Varying Call Sizes

It was observed that Library 1 monitored the call sizes and reconfigured the video in order to reduce bandwidth usage in the case of multi-user calls. Video from DUT 2 was observed to consume a lot of bandwidth and as expected, evidence of traffic shaping was visible for DUT 2. When the call size increased from 2 to 3, the video was reconfigured to occupy less bandwidth. As evident from the available frame sizes, the reconfigured video has a bit rate of 256 Kbps.

7.4 Summary

The important conclusions derived from the above sections can be summarised as follows:

1. Both the libraries try to maintain a near constant frame rate and use frame sizes to throttle the sending rate.

2. Library 1 reacts to packet loss by increasing the source rate. The increase in source rate is due to increase in frame sizes. The increase in frame sizes is partly due to

the presence of more number of I frames in the trace. Thus even though error control is seen, the video quality is badly affected due to packet loss.

3. Library 1 reacts to increasing call sizes by reconfiguring the video to occupy less bandwidth.

4. Library 1 is not sensitive to bandwidth variations and does not reconfigure the video for lower bandwidth usage even when less bandwidth is available.

5. Library 1 favours video quality over interactivity and audio.

6. Google Hangouts provides robust performance even when the available bandwidth is lower than usual. It sacrifices video quality for efficient use of bandwidth.

7. Google Hangouts prioritises audio over video and ensures that enough bandwidth is available for an audio call even if it means encoding video of a lower quality.

8. Google Hangouts prioritises interactivity over video quality and sacrifices video quality in order to improve interactivity.

In the end, it can be concluded that by use of run-time profiling, the framework can successfully extract significant information about the quality of service of the library. This information can be successfully utilised to predict the quality of experience offered by the library.

7.5 Evaluations and Future Recommendations

It was observed that for ideal testing of video conferencing libraries, a common video source for all experiment sets should have been used. However as that was not possible owing to limitations in the available software, two sets of video sources have been used:

1. Camera facing plain paper (resulting in omission of random effects arising out of motion).

2. Camera facing normal surroundings (single person in front of camera) where motion, although unreliable is limited.

However, in the future, a standardised video source can be used resulting in standardised and more reproducible results.

Jitter Emulation

Jitter is recognised as of the main identifiers which can be used to detect, measure and/or model congestion. The framework currently does not emulate jitter and in the future, jitter emulation can be added to measure the library's response.

Packet Loss

In the case of packet loss, it is difficult to estimate loss of quality by loss of packets as not all packets are equally important to video/audio quality. In the future, video quality can be measured comprehensively bVQM [2] or other similar algorithms and the loss in quality due to packet loss can be accurately measured and quantified.

Topology

The framework is currently optimised for the following topologies:

1. A peer-to-peer topology for two users.
2. A client-server topology for two or more users.

A fully meshed peer-to-peer topology for a multi-user conference is not yet tested using the framework and some modifications will be needed in this particular use case.

Automation

Real-Time Transport protocol is used by most libraries for real-time transport. Wireshark can be made to do an even more detailed statistical analysis by carefully decoding and filtering various components of the RTP stream (for e.g. different RTCP feedback messages, statistics like ratio of audio to video bandwidth usage etc.). The accuracy and relevance of the framework would improve even more if the framework does this analysis automatically and generates highly relevant information for user interpretation.

Bibliography

[1] *2013 video chat im software product comparisons.*

[2] *Batch video quality metric (bvqm) user's manual.*

[3] *Cisco telepresence secure communications and signaling.*

[4] *Cisco white paper on video conferencing standards.*

[5] *Farstream - audio/video communications framework.*

[6] *Goldwave.*

[7] *Ip geolocation website.*

[8] *Msu video quality measurement tool.*

[9] *Presence displays bv.*

[10] *Web real time communication.*

[11] *Polycom high-definition (hd) video conferencing*, 2005.

[12] *Usability, accessibility, research and consulting*, 2013.

[13] Acrobits, *Acrobits mobile voip solutions.*

[14] Andbang, *Andbang.*

[15] Android, *Acousticechocanceler.*

[16] AOL, *Aol instant messaging.*

[17] Apple, *icall.*

[18] Auralink, *Auralink.*

[19] Avaya, *radvision an avaya company.*

[20] Salman A Baset and Henning Schulzrinne, *An analysis of the skype peer-to-peer internet telephony protocol*, arXiv preprint cs/0412017 (2004).

[21] Mark Baugher, D McGrew, M Naslund, E Carrara, and Karl Norrman, *The secure real-time transport protocol (srtp)*, 2004.

[22] BDTI, *How video compression works*, 2007.

[23] Scott Bradner, *The internet engineering task force*, Open sources: Voices from the open source revolution (1999), 47–52.

[24] Ben Campbell, Jonathan Rosenberg, Henning Schulzrinne, Christian Huitema, and David Gurle, *Session initiation protocol (sip) extension for instant messaging*, (2002).

[25] Stephen L Casner, *Media type registration of rtp payload formats*, (2007).

[26] Kuan-Ta Chen, Chun-Ying Huang, Polly Huang, and Chin-Laung Lei, *Quantifying skype user satisfaction*, ACM SIGCOMM Computer Communication Review, vol. 36, ACM, 2006, pp. 399–410.

[27] M Reha Civanlar, Öznur Özkasap, and Tahir Çelebi, *Peer-to-peer multipoint videoconferencing on the internet*, Signal Processing: Image Communication **20** (2005), no. 8, 743–754.

[28] Apple Computers, *Apple ichat*, 2013.

[29] Counterpath, *Counterpath*.

[30] Luca De Cicco, Saverio Mascolo, and Vittorio Palmisano, *Skype video responsiveness to bandwidth variations*, Proceedings of the 18th International Workshop on Network and Operating Systems Support for Digital Audio and Video, ACM, 2008, pp. 81–86.

[31] Ekiga, *Ekiga*.

[32] K El-Khatib, G Luo, G Bochmann, and F Pinjiang, *Multiplexing scheme for rtp flows between access routers*, INTERNET ENGINEERING TASK FORCE (IETF), INTERNET DRAFT **24** (1999), 1–13.

[33] Eyeball, *Eyeball chat*.

[34] Roy Fielding, Jim Gettys, Jeffrey Mogul, Henrik Frystyk, Larry Masinter, Paul Leach, and Tim Berners-Lee, *Hypertext transfer protocol–http/1.1*, 1999.

[35] Frank HP Fitzek and Martin Reisslein, *Mpeg-4 and h. 263 video traces for network performance evaluation*, Network, IEEE **15** (2001), no. 6, 40–54.

[36] Sally Floyd and Kevin Fall, *Promoting the use of end-to-end congestion control in the internet*, IEEE/ACM Transactions on Networking (TON) **7** (1999), no. 4, 458–472.

[37] Google, *Google hangouts*.

[38] Christina Hatting et al., *End-to-end qos network design*, Cisco Press, 2005.

[39] Stephen Hemminger et al., *Network emulation with netem*, Linux Conf Au, Citeseer, 2005, pp. 18–23.

[40] Bert Hubert, Thomas Graf, Greg Maxwell, Remco van Mook, Martijn van Oosterhout, P Schroeder, Jasper Spaans, and Pedro Larroy, *Linux advanced routing & traffic control*, Ottawa Linux Symposium, 2002, p. 213.

[41] i-p-tel GmbH, *Free sip/voip client.*

[42] Van Jacobson, Craig Leres, and S McCanne, *The tcpdump manual page*, Lawrence Berkeley Laboratory, Berkeley, CA (1989).

[43] Van Jacobson, Craig Leres, and Steven McCanne, *pcap-packet capture library*, UNIX man page (2001).

[44] Cullen Jennings and Suhas Nandakumar, *Sdp for the webrtc*, (2013).

[45] On2 Jim Bankoski, *The vp8 video codec: High compression+low complexity.*

[46] Jitsi, *Jitsi.*

[47] Jukka Korpela, *What rfcs are.*

[48] KPhone, *Kphone.*

[49] HTML5 Labs, *Interoperability cu-rtc-web*, February 2013.

[50] Ulf Lamping and Ed Warnicke, *Wireshark user's guide*, Interface **4** (2004), 6.

[51] LifeSize, *Lifesize softphone.*

[52] LinPhone, *Linphone.*

[53] Yue. Lu, Fernando. Kuipers, Yong. Zhao, and Piet Van Mieghem, *Optimizing polynomial expressions by algebraic factorization and common subexpression elimination*, Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on (2010), 96–107.

[54] Michael Luby, Lorenzo Vicisano, Jim Gemmell, Luigi Rizzo, M Handley, and Jon Crowcroft, *Forward error correction (fec) building block*, Tech. report, RFC 3452, December, 2002.

[55] Rohan Mahy, Philip Matthews, and Jonathan Rosenberg, *Traversal using relays around nat (turn): relay extensions to session traversal utilities for nat (stun)*, Internet Request for Comments (2010).

[56] Niall Mansfield, *Practical tcp/ip: designing, using, and troubleshooting tcp/ip networks on linux and windows*, Addison-Wesley, 2003.

[57] Jan Mastalir, *Understanding sip-based voip*, 2013.

[58] microsip, *microsip.*

[59] Microsoft, *Skype.*

[60] OOVOO, *Oovoo.*

[61] OPAL, *Opal voip solutions.*

[62] OpenTok, *Getting started with opentok 2.0.*

[63] Angela Orebaugh, Gilbert Ramirez, and Jay Beale, *Wireshark & ethereal network protocol analyzer toolkit*, Syngress, 2006.

[64] Joerg Ott and Varun Singh, *Evaluating congestion control for interactive real-time media*, (2013).

[65] Shrideep Pallickara and Geoffrey Fox, *Naradabrokering: a distributed middleware framework and architecture for enabling durable peer-to-peer grids*, Middleware 2003, Springer, 2003, pp. 41–61.

[66] PJSIP, *Pjsip version 2.1.*

[67] Plivo, *Voip sdk - add voice communications to web and mobile apps.*

[68] QuteCom, *Qutecom.*

[69] Revation, *Revation.*

[70] Janko Roettgers, *The technology behind google+ hangouts*, June 2011.

[71] Jonathan Rosenberg, Henning Schulzrinne, Gonzalo Camarillo, Alan Johnston, Jon Peterson, Robert Sparks, Mark Handley, Eve Schooler, et al., *Sip: session initiation protocol*, Tech. report, RFC 3261, Internet Engineering Task Force, 2002.

[72] Henning Schulzrinne, *Rtp: A transport protocol for real-time applications*, (1996).

[73] Heiko Schwarz, Detlev Marpe, and Thomas Wiegand, *Overview of the scalable video coding extension of the h. 264/avc standard*, Circuits and Systems for Video Technology, IEEE Transactions on **17** (2007), no. 9, 1103–1120.

[74] Stage SDP, *Session initiation protocol*, (2002).

[75] Skype, *Skype - free im & video calls.*

[76] William Stallings, *Handbook of computer-communications standards; vol. 1: the open systems interconnection (osi) model and osi-related standards*, Macmillan Publishing Co., Inc., 1987.

[77] Hadeel H Taha, E Koerner, and K Reinhardt, *Architecture for a sip-based conferencing server*, Computer Engineering Department, University of Applied Sciences Mannheim (2005).

[78] Gary A Thom, *H. 323: the multimedia communications standard for local area networks*, Communications Magazine, IEEE **34** (1996), no. 12, 52–56.

[79] TokBox, *Tokbox.*

[80] tringme, *Tring me.*

[81] Thierry Turletti, *H. 261 software codec for videoconferencing over the internet*, (1993).

[82] Vidyo, *Vidyo*.

[83] W3C, *Web real-time communications working group*.

[84] Samsung Website, *Samsung exynos 5 dual*.

[85] Elias Weingartner, Hendrik Vom Lehn, and Klaus Wehrle, *A performance comparison of recent network simulators*, Communications, 2009. ICC'09. IEEE International Conference on, IEEE, 2009, pp. 1–5.

[86] Stephan Wenger, U Chandra, M Westerlund, and B Burman, *Codec control messages in the rtp audio-visual profile with feedback (avpf)*, Work in Progress (2007).

[87] Paul Wilkins, *The on2 vp6 codec*.

[88] MH Willebeek-LeMair, Dilip D Kandlur, and Z-Y Shae, *On multipoint control units for videoconferencing*, Local Computer Networks, 1994. Proceedings., 19th Conference on, IEEE, 1994, pp. 356–364.

[89] Dapeng Wu, Yiwei Thoms Hou, and Ya-Qin Zhang, *Transporting real-time video over the internet: Challenges and approaches*, Proceedings of the IEEE **88** (2000), no. 12, 1855–1877.

[90] Yang Xu, Chenguang Yu, Jingjiang Li, and Yong Liu, *Video telephony for end-consumers: Measurement study of google+, ichat, and skype*, Proceedings of the 2012 ACM conference on Internet measurement conference, ACM, 2012, pp. 371–384.

[91] Yahoo, *yahoo messanger*.

[92] Xinggong Zhang, Yang Xu, Hao Hu, Yong Liu, Zongming Guo, and Yao Wang, *Profiling skype video calls: Rate control and video quality*, INFOCOM, 2012 Proceedings IEEE, IEEE, 2012, pp. 621–629.

[93] Xinyan Zhang, Jiangchuan Liu, Bo Li, and Y-SP Yum, *Coolstreaming/donet: a data-driven overlay network for peer-to-peer live media streaming*, INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE, vol. 3, IEEE, 2005, pp. 2102–2111.

[94] Robert Zopf, *Real-time transport protocol (rtp) payload for comfort noise (cn)*, (2002).